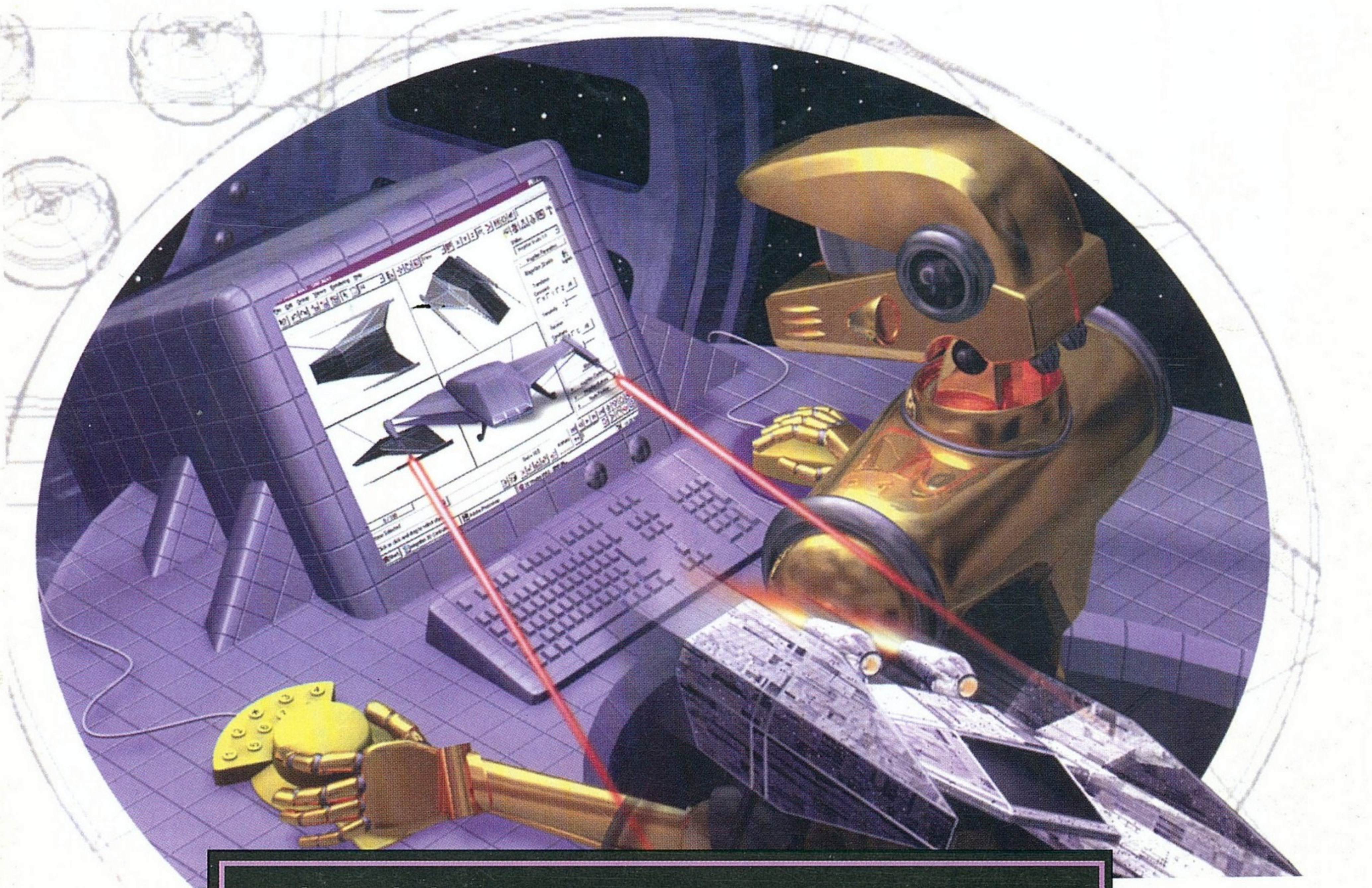
XVI. ÉVFOLYAM 5. SZÁM, 1998. MÁJUS, ÁRA: 588 FT

ÚJ ALAPLAP

MAGYAR SZÁMÍTÁSTECHNIKAI FOLYÓIRAT



A HÓNAP TÉMÁJA:

TÁRGYSZERŰSÉG

A CD-mellékleten:
CorelDraw8 próbaverzió

Fogódzó:
Hardverleíró világnyelv

Pro domo:
8 év után búcsú a floppytól

Vírusőrjárat:
Cap, a szürke kis karrierista


KÖSSÖN
BELÉNK...
... bérelt vonalon.
Az Elender Informatikai Rt.
különleges lehetőséget kínál
vállalati felhasználóknak:
Új bérelt vonali Internet előfizetés esetén
a szolgáltatás igénybevételéhez szükséges
CISCO routert ingyenesen biztosítja.
Üzleti kommunikációs szolgáltatásainkról és az Ön cégének legmegfelelőbb és
legtakarékosabb Internet kapcsolat kialakításának lehetőségeiről készséggel
tájékoztatjuk Önt bővebben a 465-7867-es telefonszámon.
elender internet

ÚJ ALAPLAP

A Mikroszámítógép Magazin és az Alaplap hagyományait folytató magyar számítástechnikai folyóirat

Megjelenik havonta CD-melléklettel

Főszerkesztő:
Faklen Pál

Főszerkesztő-helyettes:
Varga János

Szerkesztő:
Jakab Ágnes

A szerkesztőbizottság tagjai:
Ambrózy Gábor, Aszalós László, Feleki Zoltán, Galántai Zoltán, Herczeg József, Horlai János, Kis János, Kovács István, Mózes István Miklós, Pogány Csaba, Simay Endre István, Szondi Egon János, Vargha Dénes, Vékony Tamás

Szerkesztőség és kiadó:
1539 Budapest, Pf. 571
VI., Dózsa György út 84/b
Telefon: 322-4417, 322-5238
Fax: 351-8015
E-mail: alaplap@mail.datanet.hu

Felelős kiadó:
Faklen Pál

Terjesztés:
Megyes Zsuzsanna

Hirdetésszervezés:
Árvai Katalin, Bogácsi Mária, Galyasi Hedvig, Pap Katalin

Külföldi hirdetések:
PubliCity
Reklám- és Médiaügynökség
1537 Budapest I., Márvány u. 17.
Telefon: 156-1182 Fax: 175-3539

A kiadó a hirdetések tartalmáért és a nyomdakészen kapott hirdetések formájáért (és helyesírásáért) nem vállal felelősséget

Példányszámadatok hitelesítése:
Magyar Terjesztésellenőrző Szövetség
MATESZ
Ez a szám 10 000 példányban jelent meg

Nyomtatás:
Zalai Nyomda Rt, Zalaegerszeg
Felelős vezető:
Czirkl György vezérigazgató

Terjeszti:
A Magyar Posta Rt, a Nemzeti Hírlapkereskedelmi Rt, a Hírker Rt, a Kiadói Lapterjesztő Kft és számos számítástechnikai szaküzlet

Előfizethető a kiadónál:
Új Alaplap Kiadói Kft, 1539 Budapest, Pf. 571
Bankszámlaszám:
OTP 11706016-20788599

A lap példányonkénti ára: 588 Ft
Évi előfizetési díj: 5880 Ft

Külföldi előfizetés díja:
5880 Ft + postázási költség

HU ISSN 1217-7598

XVI. ÉVFOLYAM 5. SZÁM, 1998. MÁJUS

TARTALOM

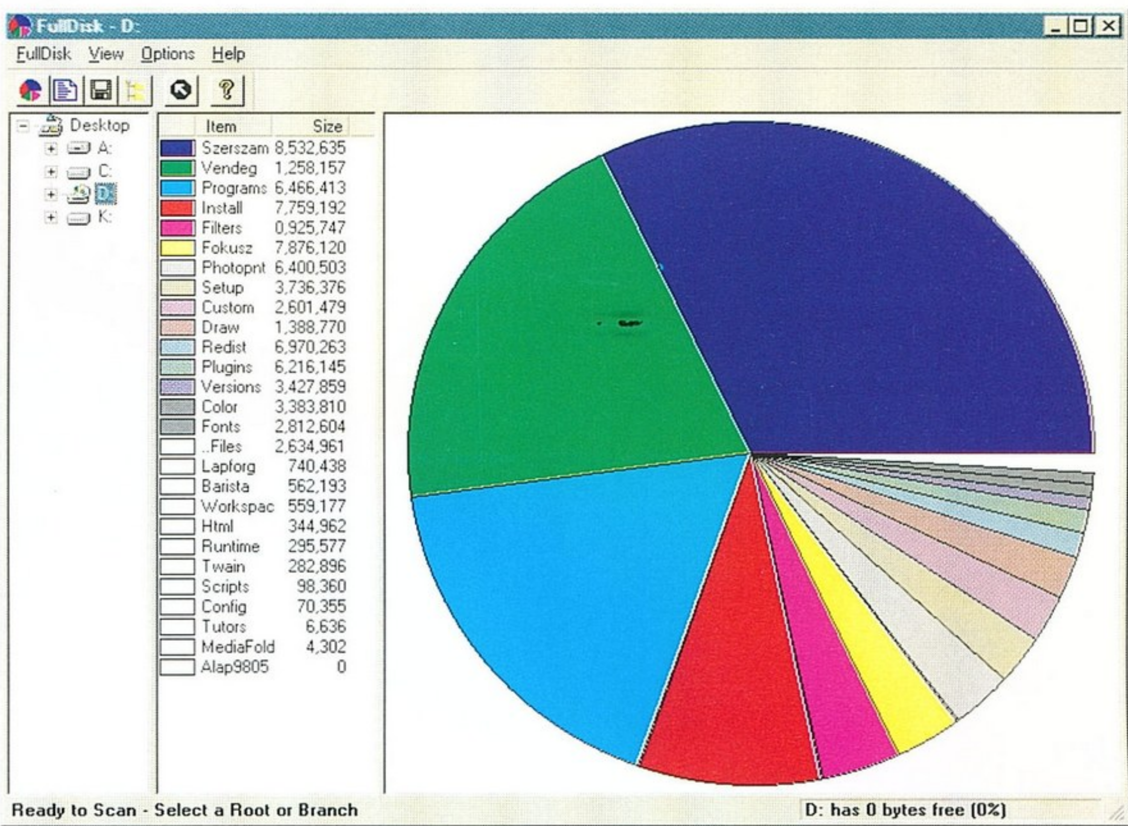
Az Új Alaplap CD-mellékletének tartalma — 1998. május

ALAP9805	**A májusi floppymelléklet**
HTML	**A CD HTML-felülete**
INSTALL	**Telepítőanyagok platformok szerint**
FOKUSZ	**Fókusz rovat**
DELPHI	— Delphi komponensek
	Lapraforgó rovat
LAPFORG	— Képek a Photoshop sulihoz
RUNTIME	— Telepítéshez szükséges kiegészítő anyagok
SZERSZAM	**Szerszámosláda rovat**
DOSWIN	— DOS és Win 3.1-es shareware-gyűjtemény
LINUX	— Linux-anyagok
OS2	— OS/2-válogatás
WIN95	— Win95-ös gyűjtés
WINNT	— Windows NT-programok
VENDEG	**Vendégoldal rovat**
ARTISOFT	— Lantastic 7.0 demó és kipróbálási változat
ELENDER	— Elender internetes starter kit
MPCOMP~9	— Az MP Computer bemutatója: Netdemo



A CorelDraw 8 bemutató változata (30 napos)

BARISTA
COLOR
CONFIG
CUSTOM
DRAW
FILTERS
FONTS
MEDIAFOLDERS
PHOTOPNT
PLUGINS
PROGRAMS
REDIST
SCRIPTS
SETUP
TUTORS
TWAIN
VERSIONS
WORKSPACE



Az Új Alaplap floppymellékletének tartalma — 1998. MÁJUS

Faklen Pál	Búcsú a floppytól — BUCSU.TXT	➾61. o.
Magyar Tamás	Fájlinfó Windowshoz — FINFO.TXT, FINFO#.EXE	
Kálmán Károly	Gyorsfénykép az XMS-ről — XMSINF.TXT, XMSINF.EXE	
Kiss J. Gábor	Különleges floppyformátumok — \ALAP9805\FLOPFORM*.*	
Borsi Zsolt — Simon László	„Szerkeszthető” Európa — EUROP#.EXE	
Nagy Gyula	PC Rébusz 98/05 — PCR.TXT, PCR9805.EXE, F*.XXX	

Objektumorientáltan

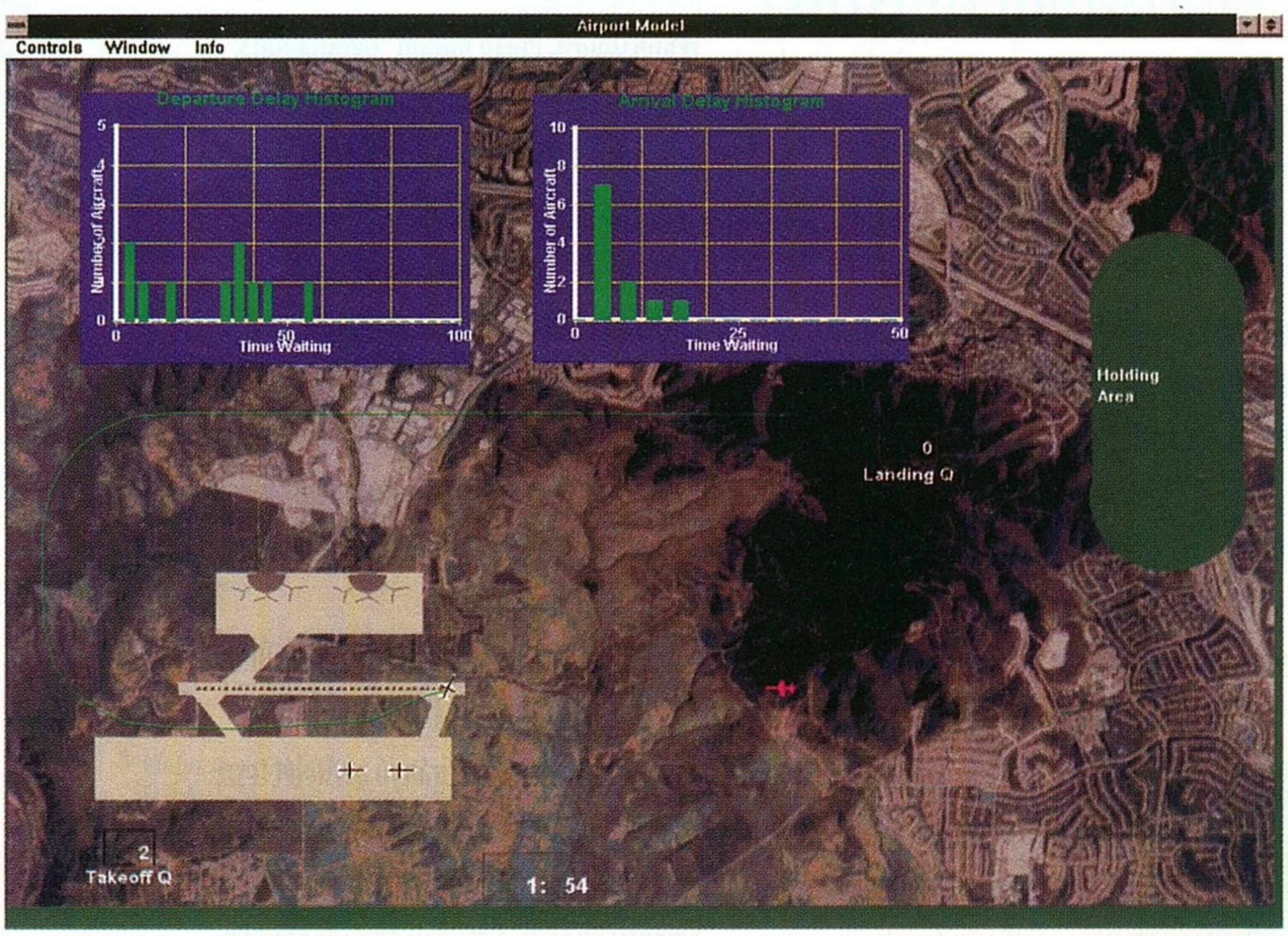
A számítástechnika rövid, de eseményekben gazdag történetének kevés tendenciája váltott ki annyi ellentétes értékelést, mint a szemünk láttára most kibontakozó nagy fellendülés, egyelőre főleg csak a nagy szoftverrendszerek fejlesztése terén. Sokak szerint ugyanis az objektumorientált programozásban sikerült megtalálni azt a csodagyógyszert, amellyel a fenyegető szoftverkrízist le lehet küzdeni. Mások ezzel szemben arra figyelmeztetnek, hogy csak ne nagyon lelkendezzünk, mert ez a fejlődési irányzat rengeteg veszélyt is hordoz magában.

Kétségtelen, hogy a hardvereszközök hihetetlen tempójú tartós fejlődését sem szakmai, sem pénzügyi eszközökkel nem lehet visszafogni. A hardverkínálat sok szempontból messze előtte jár a valóságos szükségleteknek. Semmi baj — mondják a hardvergyártók —, az igényeket mesterségesen is feljebb lehet tornázni. Ehhez viszont szoftverek kellenek, mégpedig olyanok, amelyek mindig a fejlődés adott szintjéhez alkalmazkodnak. Kulcskérdés tehát az, hogy miként lehet biztosítani a hardver fejlődésével való lépéstartást a szoftverfejlesztésében, amikor a hagyományos módszerek nem elégségesek.

Az igényformálás hatásos eszközének bizonyult a grafikus kezelői felület, leginkább pedig az MS Windows. A látványos képi eszközökkel operáló egységes programkörnyezet olyan alkalmazásoknál is túlsúlyba került, ahol valójában semmi keresnivalója nem lett volna, hiszen a karakteres megoldás olcsóbb is, gyorsabb is, biztonságosabb is... A kitűnő piacbefolyásoló technika révén viszont sikerült levezetni legalább egy hardvergeneráció minőségi túltermelését. A gépek képességeinek túlnyomó részét manapság már nem a valóságos feladat megoldására használjuk, hanem a látvány előállítására, képek és hangok együttesének lenyűgöző megjelenítésére, a film- és videotechnika által jóval korábban alkalmazott trükkök digitális alapokon történő leutánzására.

A fejlődés előrehaladásán azonban kár volna siránkozni. Inkább azon kell ügyködnünk, hogy a táguló lehetőségeket próbáljuk meg értelmes módon is kihasználni, hiszen számos olyan terület van, amelynek indokolt és érdemi számítástechnikai igényeit a jelenlegi eszközökkel még nem tudjuk kielégíteni. Ilyen például az írott formában szunnyadó hatalmas tudásanyag hozzáférhetővé tétele, ilyen az emberi beszéd online feldolgozása és fordítása... Ma még megcsodáljuk, ha gazdája szavára egy autó leengedi az ablakot vagy kikapcsolja a világítást, és inkább azt érezzük természetesnek, hogy a számítógéppel történő kommunikáció mindkét kezünket lefoglalja.

Tennivalónk van elég, és marad belőle utódainknak is. A kérdés csak az, hogy mikor milyen módszerekkel, megoldásokkal tudunk eleget tenni a kihívásoknak — és hogy értelmesen használjuk-e fel azokat az eszközöket, amelyek rendelkezésünkre állnak. E havi témánk, az objektumorientáltság, azon belül is az OOP szintén csak az egyik ilyen eszköz, de hogy milyen hatalmas lehetőségek rejlenek benne, azt most kezdjük igazán megérteni. Aki egyelőre félvállról veszi, vagy úgy érzi, hogy a téma „nem rá tartozik", az legalább hallgassa meg, mit mondanak olyanok, akik már találkoztak vele. Érdemes odafigyelni a szavukra.



Szubjektív vallomás

Tapasztalatok OO „körökben”

Cikkünk szerzője 1988-ban — franciaországi ösztöndíja idején — még csodabogárnak nézte azt a svéd fiút, aki az időjárási és agrárgazdálkodási folyamatok objektumorientált modellezésével töltötte az idejét (C++ és Simula nyelven) — ahelyett, hogy rögtön nekilátott volna programjai megírásának.

Ma már elmondható, hogy az objektumorientált (a továbbiakban: OO) technológia az információs korszak általánosan alkalmazott szoftverfejlesztési módszere. Az elmúlt 6-8 év alatt számos szakember teljesen elkötelezte magát az objektumorientált programozási módszer (OOP) mellett. Velem is ez történt. Franciaországban a svéd fiúval való megismerkedés felkeltette az érdeklődésemet, de akkor még azt hittem, hogy az OO csupán túlzott intellektussal bíró pihentagyúak szórakozási formája. Azután 1989-ben kezembe került a Borland Turbo Pascal 5.5 leírása az OOP-ről. Harmadik olvasásra kezdett megvilágosodni a lényeg: csakugyan ez az, ami nekem kell, ez az a módszer, amely tényleg megnövelheti a szoftverek készítésének hatékonyságát. Nemcsak érteni, hanem érezni is kezdtem...

1995-96-ban, ismét Franciaországban, az objektumorientált tervezésbe (OOT) is beletanultam. Életem első objektum-adatbáziskezelőjét, az ObjectStore-t, 1996-ban ismerhettem meg, ezt követte később még néhány (Versant C++, GemStone/J, POET C++ és Java). 1997-ben már a Java programozás és eszközök, valamint az elosztott (Corba) objektumok világában is el tudtam mélyedni.

Ma már elmondhatom, hogy az OO technológia valamennyi területén részt vettem termelésben működő rendszerek létrehozásában a tervezéstől a programozásig (C++-ban és Javában), az adatbáziskezeléstől objektumkönyvtárak tervezéséig és készítéséig. Összegyűlt tapasztalataim alapján megpróbálok segíteni azoknak, akik szeretnének eligazodni ezen az igencsak bozótos területen.

Két éve — akkor már az IQSoftnál dolgoztam — létrehoztuk az Objektum Orientált Fejlesztők és Felhasználók Klubját (OOFFK) a Neumann János Számítógéptudományi Társaság egyik szakosztályaként. Havonta tartunk szakmai összejöveteleket, és azóta már két országos konferenciát is szerveztünk (1996-ban Kecskeméten, 1997-ben Visegrádon).

Objektumorientált tervezés

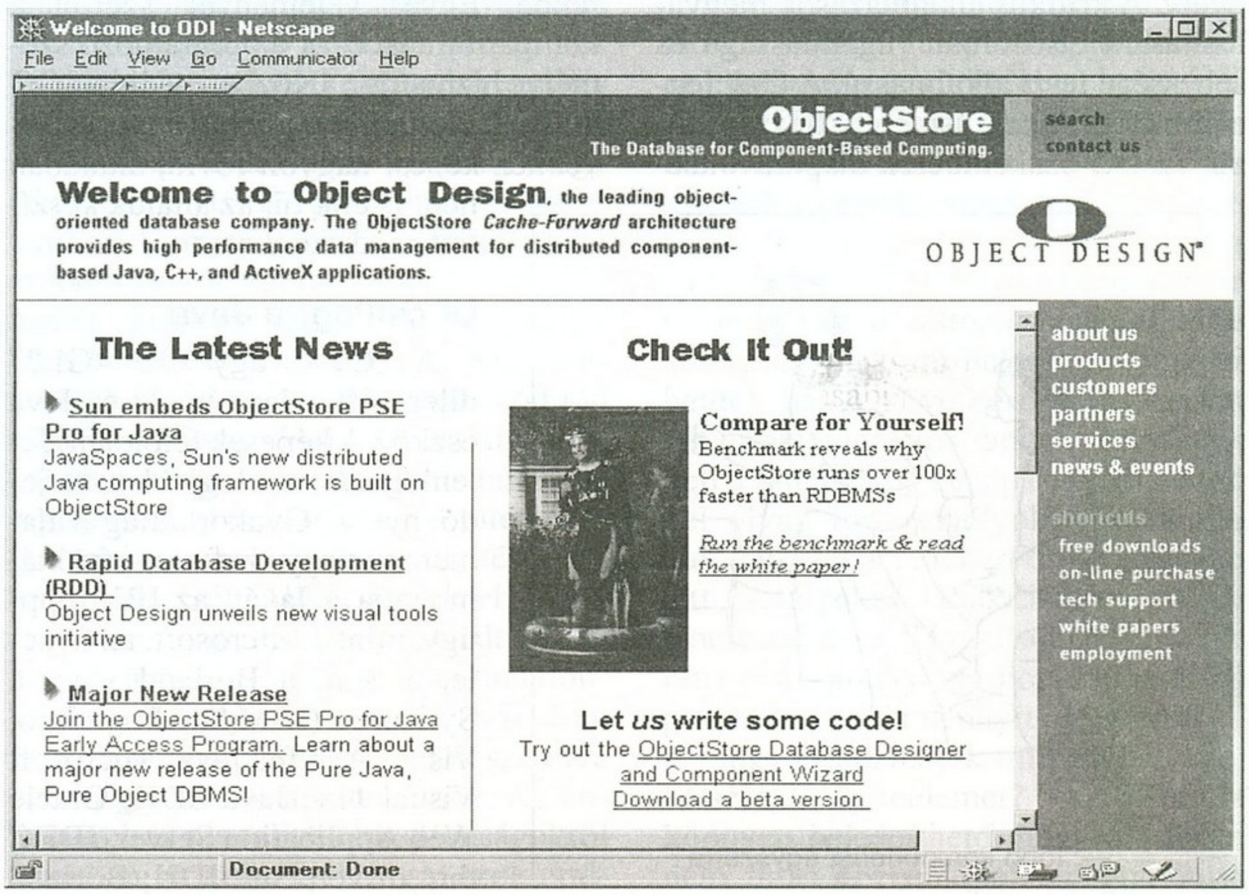
Az OOT ennek az évtizednek a találmánya. 1989 és 1994 között megszületett az összes lényeges (elemzési és) tervezési módszertan, amelyek közül a legnagyobb népszerűségre az 1991-ben publikált [1] Object Modeling Technique (OMT) tett szert (James Rumbaugh és társai).

Az OO módszerek között nincs lényeges eltérés. Az első OO tervezési módszer megtanulása elsősorban szemléletváltozást követel meg a tervezőtől. Aki még nem programozott OO módszerrel, annak számára nagy eltökéltséget igénylő feladat a hatékony OOT-hez szükséges szakértelem megszerzése. Az OOT elveit azonban igazán csak a gyakorlatban, az OOP technikáin keresztül lehet megtanulni. Kijelentettem, hogy az OO rendszertervek sokkal gyakorlatiasabbak, mint a strukturált módszerrel készült rendszertervek, mivel kifejezetten az előállítandó szoftverre vonatkoznak.

Az OOT módszerei fokozatosan átveszik a strukturált tervezési módszerek helyét. 1996-ban már az újonnan indított szoftverfejlesztési projektek többségénél OOT módszertant alkalmaztak. (Természetesen ez csak azokra a fejlesztésekre vonatkozik, amelyekben formális a tervezési módszer.) Magyarországon ez a fordulat alighanem 1998-ban fog bekövetkezni.

Az OOT előretörését és sikerét elsősorban nem magában az OO tervezési módszerben kell keresni, hanem az OO programozási nyelvek (C++, Java) és eszközök (Visual Basic 5, Centura, PowerBuilder, Forté) elterjedésében. Olyan szoftvert, amelyet OO nyelven fognak implementálni, sokkal jobban lehet OO módszerrel tervezni. Az OOT elsősorban a részletes rendszerterv elkészítésében jelent segítséget.

Főleg a bezárás (encapsulation) és az interfészek (design by contract) fogalmának támogatása teszi alkalmassá az OO fejlesztési módszert (fejlesztés = tervezés + programozás) iteratív jellegű, gyors, prototípus-vezérelt fejlesztések elvégzésére is.



Az OOT egyik legnagyobb előnye, hogy megjelentek kifejezetten az újrafelhasználhatóságot támogató komponensek (JavaBeans, ActiveX). Ezek használata ma már teljesen általánossá vált, hiszen jelentősen csökkentik a fejlesztési költségeket.

A három legelterjedtebb OOT módszer az OMT, a Booch és az Objectory. E módszerek szerzői (J. Rumbaugh, G. Booch és I. Jacobson) dolgozták ki a Unified Modeling Language (UML) jelölésrendszert, amelyet az Object Management Group (OMG) szabványként is elfogadott. A legnagyobb szoftverházak (IBM, Microsoft, Oracle stb.) ma már integrálják eszközeikbe az UML-t. (Az UML-ről bőséges irodalom található, főleg a [2] és [3] könyveket ajánlom az olvasók figyelmébe.)

Az OO elemzés területén is elkészültek már a legfontosabb módszerek ([4], [5]), ezek gyakorlatban való hatékony alkalmazása jelenleg az egyik legizgalmasabb kihívás a rendszerszervezők számára.

Objektumorientált programozás

Az OO technológia legfontosabb (központi) része az objektumorientált programozás; minden más OO területet az OOP mozgat. Az OOT az OOP fogalmait, elveit általánosította a tervezésre. Az OO adatbázisok az OO programok objektumait tárolják olyan formában, ahogy azok a programokban megjelennek. Az elosztott komponenseket az jellemzi, hogy ezek C++ vagy Java objektumokként valósulnak meg az alkalmazásszervereken.

Az OOP legmeghatározóbb és (jelenleg még) legelterjedtebb tagja a C++ nyelv. A kritikus alkalmazások megvalósítására elsősorban rugalmassága és sebessége teszi alkalmassá. A C++ legalább nagyvonalú ismerete hozzátartozik az OO-szakemberek alapműveltségéhez. A C++ azonban nem könnyű nyelv, ezért számos próbálkozás történt egyszerűbb nyelvek alkalmazására az OOP céljaira. Az óriási irodalomból kettőt említek meg az irodalomjegyzékben ([6], [7]), de mindkettő csak haladóknak vagy szuperhaladóknak való. A C++ rendszerek tervezéséről talán Booch [8] írta le a legmélyebb gondolatokat.



— Több komponenst egyszerre?

4G fejlesztőeszközök: tündöklés és bukás

Az OOP-t igazából nem a C++ tette általánosan elterjedtté, hanem az OO és 4. generációs (4G) fejlesztőeszközök, a Centura, a PowerBuilder, a Forté és egyebek. Ma már a Visual Basic 5-ös változata is nyújt OOP-t támogató szolgáltatásokat. Relációs (SQL) adatbázisokon nyugvó üzleti alkalmazások kifejlesztéséhez ezek az eszközök sokkal termelékenyebb környezetet nyújtanak, mint amelyekben szükség van a C++ programozásra. Az utóbbi időben a legnagyobb rendszerek ilyen OO 4GL-ekkel készültek.

A legnagyobb baj ezekkel a rendszerekkel az, hogy komponenseik más rendszerek számára nem elérhetőek (egy PowerBuilder objektumgyűjtemény a Centura projektek számára nem elérhető, és fordítva). Az egyetlen lehetőség az együttműködésre a relációs adatbázisok tábláinak, tárolt eljárásainak szintjén jelenik meg. Kivétel nélkül valamennyi OO 4GL-ről elmondható, hogy önmagán kívül semmi mással nem kompatibilis, sem a nyelve, sem a programozói környezete. Az elkészült program forráskódjának a legkisebb része sem vihető át közvetlenül más rendszerekbe. Egyes vélemények szerint a kompatibilitást csak a programozó személye biztosítja... Közben pedig az OO 4GL-ek élettartama a szabványos nyelvekhez képest nagyon rövid, általában nem is élik túl az általuk készített rendszereket.

Új csillag: a Java

A „C++ vagy OO 4GL?" dilemmát a Java nyelv és Java eszközök képesek feloldani. Jelenleg a Java a legjobban terjedő nyelv. Gyakorlatilag valamennyi nagy szoftvergyártó támogatja a Javát, az IBM éppúgy, mint a Microsoft, az Oracle, a Sun, a Borland vagy a Symantec (IBM San Francisco, Visual Age for Java; Microsoft Visual J++, Java SDK; Oracle Web Application Server, JDBC thin driver; Sun JDK; Borland JBuilder; Symantec Visual Café; stb.). Már most több eszköz áll a javások rendelkezésére, mint az összes többi nyelvben összesen.

Népszerűségben és fejlődési ütemben a Java ma már a Microsoft világát is elhomályosítja. Nem véletlenül élvez ekkora támogatást: műszaki szempontból kiváló programnyelv, ugyanakkor könnyen megtanulható. Sebességben sem maradnak el a C++ programok mögött azok a platformfüggő Java-alkalmazások, amelyeket natív fordítókkal állítanak elő (ilyen fordítókat készített a Symantec és az IBM).

A Java egyaránt alkalmas az alkalmazások kliens- és szerveroldali részeinek megvalósítására, a kliensek és szerverek közötti kommunikációban pedig — egyedülálló (dinamikus) hálózati szolgáltatásai miatt — nincs is vetélytársa. A Javában olyan szoftverek készültek el, mint a Marimba Castanet és Bongo, az ObjectSpace Voyager, amelyek szolgáltatásaikat tekintve is elképzelhetetlenek lennének Java nélkül.

Java programokat kifejezetten OOT módszerekkel lehet hatékonyan megtervezni [9]. Az OOT-ről toronymagasan a legjobb Java-könyvet P. Niemeyer és J. Peck írta [10], a legjobban használható kézikönyv pedig D. Flanagané [11].

Németh Miklós
nemeth@iqsoft.hu

SZAKIRODALOM

[1] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, F. Lorensen: Object-Oriented Modeling and Design. (Englewood Cliffs, NJ: Prentice-Hall), 1991

[2] M. Fowler, K. Scott: UML Distilled. Applying the Standard Object Modeling Language. (Reading, MS: Addison Wesley Longman), 1997

[3] H. Eriksson, M. Penker: UML Toolkit. (New York, NY: John Wiley & Sons), 1998

[4] I. Jacobson, M. Ericsson, A. Jacobson: The Object Advantage. Business Process Reengineering with Object Technology. (Reading, MS: Addison Wesley Longman), 1995

[5] N. Wilkinson: Using CRC Cards. An Informal Approach to Object-Oriented Development. (New York, NY: Sigs Books), 1995

[6] S. Meyers: Effective C++. 50 Specific Ways to Improve Your Programs and Designs. (Reading, MS: Addison Wesley Longman), 1992

[7] S. Meyers: More Effective C++. 35 New Ways to Improve Your Programs and Designs. (Reading, MS: Addison Wesley Longman), 1996

[8] G. Booch: Object-Oriented Analysis and Design with Applications. (Redwood City, CA: Benjamin Cummings), 1994

[9] P. Coad, M. Mayfield: Java Design. Building Better Apps and Applets (Englewood Cliffs, NJ: Prentice-Hall, Yourdon Press), 1997

[10] P. Niemeyer, J. Peck: Exploring Java. Second Edition. (Cambridge: O'Reilly), 1997

[11] D. Flanagan: Java in a Nutshell. A Desktop Quick Reference. Second Edition. (Cambridge: O'Reilly), 1997

Zárt osztály, vigécek, dzsinnek...

OOP Delphiben

A Delphi integrált fejlesztői környezete a vizuális programkészítés kényelmét ötvözi a Turbo Pascalból megismert és megszeretett IDE környezettel. Tudván azt, hogy a Delphi teljesen objektumorientált felépítésű, sokan azt gondolják, hogy a benne elkészített program is automatikusan rendelkezni fog az objektumorientált programok minden előnyével: biztosítani fogja az áttekinthetőséget, az újrahasználhatóságot, a könnyű karbantartás és továbbfejlesztés lehetőségét. Pedig a Delphi csak támogatja, és nem automatikusan biztosítja a jó OO programok készítését.

Magasabb szintű eszközökkel ugyanúgy — ha nem jobban — készíthetők áttekinthetetlen, kusza, „spagettiszerű" programok, mint régen, a strukturált programozás jelszavának meghirdetése előtt. Látszatra csodálatosak, látványosak lehetnek az új programok, tele ablakokkal, kombinált (combo) listákkal, nyomógombokkal. Ezek a programok napok, sőt órák alatt elkészülhetnek — de a programrendszerek világos architektúrája nem alakul ki magától.

A Delphi programok építőkövei az ún. formok, az ablakmodellek. A Windowsból ismert téglalap alakú képernyőterületeket pillanatok alatt be lehet rendezni szabványos formátumú listaablakokkal, nyomógombokkal, el lehet készíteni az ablakoknak és a gomboknak a feliratozását, sőt megfelelő funkciókkal is könnyen fel lehet ruházni ezeket a komponenseket. A baj ott kezdődik, amikor egy projekthez (= programhoz) már számos ilyen formot elkészítettünk, és egyre gyakrabban van szükségük a formoknak egymásra: az egyik működéséhez olyan információk kellenek, amelyek egy másikon keresztül érhetők csak el. Látszólag megmenti a helyzetet, hogy telepakolhatjuk formjaink implementációit „uses" klauzulákkal. De legyünk tudatában, hogy minél több ad hoc függőséget építünk bele rendszerünkbe, annál inkább elveszítjük azt a lehetőséget, hogy tisztességesen karban lehessen azt tartani, vagy — horribile dictu — más körülmények között is fel tudjuk használni.

Magas szintű absztrakció

Az OO formok újrahasználhatóságát azzal lehet biztosítani, ha minél magasabb szintű absztrakcióval készítjük el objektumainkat. A más formok számára is látható interfészek (deklarációk) szintjén ne „drótozzuk bele" formjainkba se a konkrét inicializáló kódot, se a konkrét megvalósítás módját. Ha ugyanis ezt tesszük, akkor (1) minden használat esetén ugyanígy, külön kell megadnunk az inicializáló kódot, és (2) ha más megvalósítási módot szeretnénk választani (akár csak kipróbálásra is!), az maga után vonhatja a formot használó külső kódok átírását is. Minden ilyesminek már a form implementációs részében a helye. A deklarációs részbe mindebből legfeljebb annyi való, amennyi lehetővé teszi a konkretizálást: például választási lehetőségeket kínál fel a különböző megoldási módok között, illetve lehetővé teszi a hívó kódnak, hogy az inicializálás elvégzését futás közben, a konkrét helyzetnek megfelelően maga kezdeményezze és hajtsa végre.

Gyakran észre sem vesszük, hogy akaratlanul is függőségeket alakítottunk ki formjaink között, amikor valamilyen készen kapott absztrakt komponenst „testre szabunk", közvetlen inicializáló értékek kitöltésével, vagy valamilyen rögzített megjelenítési mód idő előtti kiválasztásával. Az általános objektumok közvetlen megváltoztatásával ugyanis megbolygatjuk a beépített függőségeket, a beléjük kódolt feltételezések rendszerét. Akkor járunk el helyesen, ha megőrizzük a felhasznált objektumok általánosságát, csupán tulajdonságok formájában adunk hozzájuk annyit, amennyi a külső kódok számára szükséges a kapcsolattartáshoz. Az objektumhoz rendelhető tulajdonságokról azonban tudnunk kell, hogy azok értékek beállítására valók. Műveleteik rendkívül szegényesek, a tulajdonságértékek írására és olvasására korlátozódnak, így minden problémát ezzel a két tevékenységgel kell megoldanunk!

Ötleteink kipróbálásához igen jó lehetőséget teremthetünk azzal, ha soha nem nyúlunk a főform (main form) eredeti kódjához, hanem mindig csak valamilyen másolatot módosítgatunk. Megtehetjük azt is, hogy elkészítjük a különböző verziók implementációit, és a „uses" klauzával egy almappából azt a változatot linkeljük hozzá a főformhoz, amelyiket éppen ki akarjuk próbálni. Így az alkalmazás konfigurálását a kód változtatása nélkül, a Delphi IDE-jéből módosítgathatjuk, variálgathatjuk.

Mi mindenre jó a polimorfizmus?

Az ötletek kipróbálásának rendkívül hasznos eszköze az OOP egyik alaptulajdonságának, a polimorfizmusnak a kihasználása. Ez teszi lehetővé, hogy formunkat absztrakt osztályként deklarálva ne is használjuk másra, csak (1) interfészként a külső kapcsolatok biztosítására, (2) az absztrakt osztályból mint alaposztályból speciális alosztályok deklarálására. Ezáltal olyan kódot lehet készíteni, amely az alaposztályból levezetett bármely formot képes használni. A főform ilyenkor csak annyit tud, hogy az általa használt form az alosztályok egyike. Nem az ő hatáskörébe tartozik az erre vonatkozó információ, ez már annak a kiválasztó kódnak a dolga, amely gondoskodik a létrehozandó alosztály tényleges kiválasztásáról.

Egy ilyen kétszintű, osztályból és alosztályokból felépülő építőelemnek az elkészítése úgy történhet, hogy (például a TPickerForm unitban) közvetlenül az implementáció kódolása elé beteszünk egy neki megfelelő osztályreferenciát (TPickerFormClass = class of TPickerForm). Ez feljogosít minket arra, hogy olyan változót hozzunk létre, amelybe objektum helyett osztályt vihetünk be. A következő lépés ezután: az implementációs unitok feladatává kell tennünk, hogy csupa egyedi nevekkel olyan formosztályokat deklaráljanak (például a TPickerFormból levezetett osztályokat TPickListként, TPickComboként és TPickRadioként konkretizálva), amelyeket hozzáadhatunk a projekthez (valamennyit egyszerre!).

Hogyan hozhatjuk működésbe ezt a kétszintű építőelemet? A főformba könnyen beleépíthető, hogy a felhasználó futás közben (például gombnyo-

mással) választhassa ki, melyik variáns működését akarja kipróbálni. Megfelelő tulajdonság hozzáadásával (és az „as" operátor felhasználásával) az is megoldható, hogy ez a tulajdonság a létrehozandó kiválasztóform típusát adja vissza. (Az adott esetben a hozzáadandó tulajdonság PickerFormClass lesz, ennek típusa pedig TPickerFormClass. Ezzel lehet visszanyerni a TPickerForm kiválasztott típusát.)

Mint látjuk, az adott módszerrel viszonylag egyszerűen megoldható, hogy a változatok kezelése kedvéért ne kelljen belenyúlnunk a főform által képviselt kódhalmazba, ehelyett csupán annyit kell tennünk, hogy opcionálisan hozzáépítjük a kiválasztott változat(ok) kódját. Bár az alternatív variánsok előzetes kódolását nem lehet megúszni, projektünk alapvető kódjának szerkezetét, áttekinthetőségét ez a legkevésbé sem zavarja.

Mire jutottunk tehát eddigi vizsgálatainkban? Azt már látjuk, hogy mit lehet tenni abban a leegyszerűsített esetben, ha minden művelet, amelyet a felhasználó kérhet, teljesíthető a főformból. A valóságban természetesen bonyolultabb a helyzet, mert a projekt futása során nem maradunk meg mindig a kiinduló állásban. Előzetes döntéseinknek megfelelően változik az alapszituáció, ahogy a projekt különböző eseményeket (egérkattintást, billentyűleütést stb.) észlel. Ezek kezeléséhez valamennyi potenciálisan előforduló eseményt deklarálnunk kell a formban, és gondoskodnunk kell a kezelésükről. A lényeg azonban ettől nem változik meg: külön-külön minden szituációban tiszta képet kell teremtenünk a magunk számára, ami pedig azzal biztosítható, hogy nem építünk be kifejlesztett formjainkba átláthatatlan külső függéseket.

Vigécek és dzsinnek

Mike Scott a közelmúltban egy rendkívül érdekes előadást tartott erről a kérdésről, amelyet azután külön kiadványként is publikáltak. Scott az OOP szemléltetésére négy különböző módszer megvalósítási módját mutatta be ugyanazon a példán.

Hogy könnyebben el lehessen igazodni, válasszunk beszélő neveket a négy módszer jellemzésére: az első legyen a Zártosztályos Módszer (ZM), a második a Vigéctáskás (VM), a harmadik a Titkosításos Módszer (TM), végül a negyedik a Dzsinn Módszer (DM). Az egyes módszerek elsősorban abban térnek el egymástól, hogy mit és hogyan tesznek bele a deklarációs részbe, és miként rejtik el a különböző információkat. A fő cél természetesen az, hogy a kód lényegében változatlan maradjon, egyedül bizonyos kódrészek hozzáillesztése legyen szükséges a kellő helyeken. Scott példái is mutatják, hogy ennek a célnak az eléréséhez jól kihasználhatók a Delphi osztályainak típusmeghatározásában rejlő finom megkülönböztetések.

A legismertebb a Private és a Public kulcsszavak szerepe az osztályok deklarációiban. A Private jelöli azokat a részeket, amelyek kívülről elérhetetlenek, és kizárólag az objektum belügyének tekintendők. Ezzel szemben a Public kulcsszóval jelölt részek kívülről is, belülről is szabadon elérhetők, ezekkel tehát csínján kell bánnunk, mert belső dolgainkat nem jó, ha közszemlére tesszük.

Ravaszabb a Protected kulcsszó szerepe. Ez olyan részeket jelöl, amelyek a külvilág számára elérhetetlenek, azonban ha saját osztályokat származtatunk a védett elemekkel rendelkező osztályokból, akkor a védett összetevők publikussá válnak.

Még különlegesebb típust jelölünk a Published kulcsszóval. Ezek az elemek ugyanúgy elérhetők, mint a publikusnak deklaráltak, mégis van egy óriási különbség a Public és a Published elemek között. Nevezetesen a Published típusjelölés arra szolgál, hogy a futás közben használt tulajdonságoknak programfejlesztési, tervezési időben is értéket lehet adni, láthatóvá lehet őket tenni. Ezeket az értékeket a rendszer egészíti ki a maga (futásidejű) típusinformációival, és futás idején gondoskodik a megfelelő értékek beállításáról. Valójában ennek a típusnak a megteremtésével fejlesztette tovább a Delphi az OOP elméletét: megteremtette annak a speciális objektumnak a fogalmát, amely egyaránt elérhető fejlesztési és futási időben. Ha pedig elérhető, akkor a képernyőn is meg lehet jeleníteni, ikon formájában fel lehet tenni a palettára, használni lehet tervezés közben, amikor igazság szerint még nem is létezik. (Ezt a lehetőséget használja ki Scott negyedik módszere, amelyet Dzsinn Módszernek neveztünk el; megdörzsöljük a csodalámpát, és megjelenik a Dzsinn, lehet neki parancsokat osztogatni.)

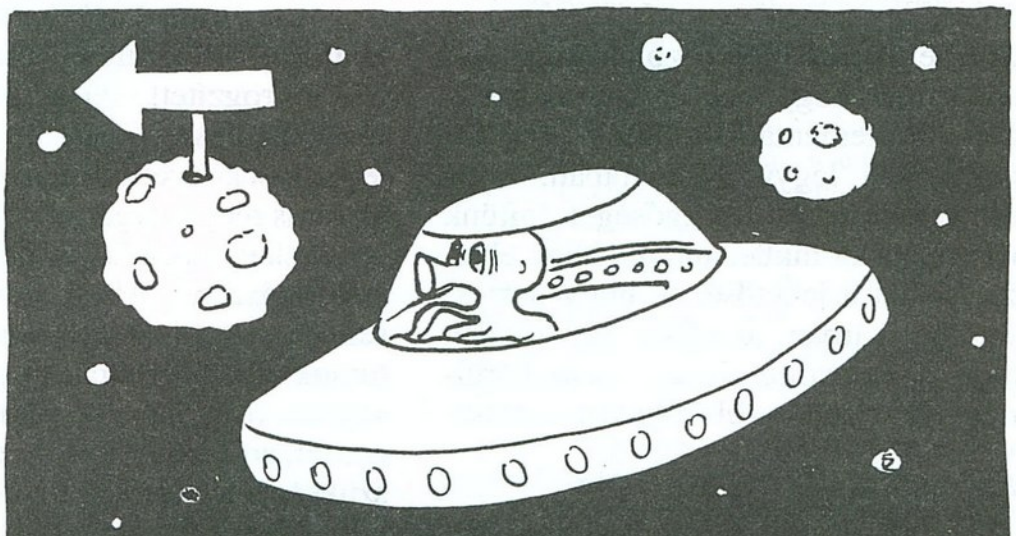
Zárt osztály

A Zártosztályos Módszer lényegében az eddig is alapelvnek tekintett „ne piszkáld!" stratégiát követi: használjuk a palettáról lehúzható kész, jól megkomponált objektumokat, és ne piszkáljunk bele a belsejükbe, mert könnyen elveszíthetjük a beléjük épített absztrakció előnyeit.

Ez a módszer a készítendő form interfészében deklarál több alprogramot (eljárást, illetve függvényt), amelyeknek az a rendeltetésük, hogy elvégezzék a működéshez szükséges tevékenységeket. Ezeket az alprogramokat azonban a ZM a deklaráción belül Protected kulcsszóval kifelé védetté teszi, sőt Virtuális minősítéssel is ellátja őket, hiszen csak futás közben derül ki, hogy mikor melyiküket kell igénybe venni. Azokat a tulajdonságokat viszont, amelyeknek a read/write műveletei megmondják, hogy az input és az output kezelésére és a választás visszaigazolására melyik függvényt vagy eljárást kell használni, a ZM módszer Public típussal jelöli, tehát közszemlére teszi.

Az elkészítendő formok nagyon hasonlóak egymáshoz, de például van köztük olyan, amely esemény segítségével tájékoztatja a főformot, ha egy másiknak a funkcióját igényli; a felhasználó a főformon gombnyomással adhatja ehhez a jóváhagyását. A lényeg az, hogy az egyes formok mindenről csak annyit tudnak, amennyi rájuk tartozik, nincs beléjük építve semmi feltételezés (például a másik működésének konkretizálásáról vagy a megvalósítás módjáról).

OOFO



OBJECT-ORIENTED FLYING OBJECT

Vigécek kíméljenek!

A Vigéctáskás (VM) módszer — amelyet egyébként angolul Service Brokernek, vagyis szolgáltatásügynöknek becéznek — események helyett mintapéldány-kollekció felkínálásával dolgozik. A Broker (a Vigéc) egy különleges objektum: bemutatja, hogy milyen változatokat lehet kiválasztani. Egy absztrakt, virtuálisnak deklarált alaposztályból indul ki az egész játék, az alaposztályhoz rendelt metódusok az interfészben mind nyilvánosak (Public típusúak). Ugyancsak nyilvános itt egy tulajdonság is, amelyet annak megfelelően állít be a főform, hogy melyik funkciót választottuk.

Ennél a módszernél a funkciók definíciójában említés sincs formról. A funkciók egyszerűen fogják a kapott adatláncot, kódjuknak megfelelően elvégeznek rajta bizonyos módosításokat, és true értékkel térnek vissza, ha működésük sikeres volt. A működés módja tehát teljes mértékben a Broker implementációjának a megoldásától függ. A megoldásnak rendszerint különböző változatai vannak, közülük a kipróbálandót egyszerűen hozzáadjuk a projekthez, anélkül, hogy bármit is módosítanunk kellene az alapkódon. A szolgáltatást végző implementációs unitnak nincs is interfésze, amelyhez bármilyen más form kívülről hozzáférhetne. A rendszer csak akkor fog kiakadni, ha elfelejtünk implementációs unitot rendelni valamelyik felkínált variánshoz.

Psszt!

A Titkosításos Módszer (TM) részben hasonló megoldásokat használ, mint a VM, de közben ugyanúgy dugdossa (Protected kulcsszóval) más formok elől a ténylegesen használt eljárásokat és függvényeket, mint a ZM, ráadásul itt Abstract metódusokként vannak deklarálva, ezeket tehát mindenképpen újra kell definiálni a származtatott osztályokban.

E módszer értelmében léteznie kell egy különálló unitnak, amelyben az alkalmazás összes elérhető osztálya absztrakt alapelemként van deklarálva. Megjegyzendő, hogy elég az elérhetőkre korlátozni az ide felvett osztályokat, hogy minimálisra szorítsuk le az absztrakt alaposztályok számát. Az absztrakt alaposztályok biztosítják a minimális interfészt ahhoz, hogy az osztályok egymást használhassák. A gyakorlatban is jól kamatoztatható az a megoldás, hogy összegyűjtjük az osztályok közötti kölcsönhatásokat, és egyetlen formban helyezzük el. Igen hasznos segédeszköz lehet ez olyankor, amikor meg akarjuk vizsgálni, hogy milyen hatással vannak egymásra a különböző osztályok.

Az implementációs kódban olyan változókat használunk, amelyek a funkciókat megtestesítő formok osztályának megfelelő típusúak. Ez az absztrakt osztály mindent magában foglal, amit a formok használatáról tudnunk kell, tehát az inputnak, outputnak és a választásnak megfelelő tulajdonságokat is. Az absztrakt interfészek egyedi implementációit form alosztályokban helyezhetjük el. Persze ezeket egyszer valahogy létre kell hoznunk, de közben szeretnénk, ha más osztályok nem is tudnának róluk. Faramuci helyzet: olyan osztályt kell létrehoznunk, amelyről nem is tudunk.

A megoldást egy különleges jogokkal felruházott Broker hozhatja meg. Ő kivételesen tudhat az alosztályokról, bár a megvalósítás módjáról nincsenek ismeretei. (A függő kód implementációs unitba van lokalizálva a vigéc számára!) A funkcióknak megfelelő formokat egy különleges típusú Broker, az ún. „osztálygyár” (ClassFactory) állítja elő, anélkül, hogy a típusukon kívül egyebet tudna róluk. (Erre jó az, hogy az implementáció külön unitba került.) Voltaképpen hasonló elv alapján működik az OLE IClassFactory interfésze is, csak az nem szabadon deklarálható, hanem rögzített típusú „osztálygyár”.

Elég különös a megoldásban az is, hogy a TClassFactoryImpl osztály csak az implementációs részben van deklarálva, és egyéb nincs is a unit interfészében. Hogy lehet egy olyan osztályt használni, amely kívülről nem is látható? A trükk igazán ötletes: inicializálásként gondoskodunk róla, hogy létrejöjjön az osztálygyár alosztályának egyik példányán. Hogy ez ne vesszen el, csupán hozzá kell rendelni egy globális példányváltozóhoz. Ennek létezéséről persze előzetesen kell gondoskodnunk azzal, hogy a közös egységben deklaráljuk a globális ClassFactory objektumot.

Valójában tehát ismét csak a polimorfizmus elvének alkalmazása teszi lehetővé, hogy olyan szolgáltatásokat is felhasználjunk, amelyeknek a kódja el van rejtve előlünk.

Aladdin és a csodalámpa

A legtisztább és leginkább az OO elvekhez illeszkedő módszer az utolsó, a Dzsinn Módszer. A Delphi egyedi komponenskezelése teszi lehetővé, hogy formjainkból teljesen elkülönített, újrahasználható komponensosztályokat hozzunk létre. Ehhez olyasféle „dzsinneket” (komponensburkokat, angolul component wrappers) hozhatunk létre, amilyeneket a dialóguspalettán lehet találni a megnyitáshoz, a mentéshez, illetve a betűtípus, a szín vagy a nyomtatási jellemzők beállítására. Ezek a dzsinnek olyan tulajdonságok felett rendelkeznek, amilyenek a form interfész tulajdonságai között szerepelnek még: ezek teszik lehetővé, hogy a futáshoz szükséges értékeket már a tervezés során beállítsuk. Adjunk hozzá minden ilyen dzsinnhez egy Execute metódust. Ez kijelzi a formot, és jóváhagyásunkat kéri: ha az OK gombra kattintunk, akkor igaz értékkel tér vissza, hasonló módon, mint a Delphi dialóguskomponensei.

Maga a komponens megírása kevésbé ijesztő, mint gondolni lehetne. Kiindulhatunk egy komponens alaposztályból, amely a szokásos módon, csak éppen Published típusként deklarálja az inputtal, outputtal és a választással kapcsolatos tulajdonságokat, továbbá Public típusként, de virtuális és absztrakt kulcsszóval a Execute függvényt. Az alosztályokban természetesen override-ként szerepel az Execute függvény megfelelő változata, a kívánt működésnek megfelelően.

Ha futás közben akarjuk használni művünket, létre kell hozni egy példányát, és fel kell tölteni a form tulajdonságait a komponensben szereplő tulajdonságoknak megfelelően.

Tervezéskor annak sincs akadálya, hogy a Delphi komponenspalettájára feltegyük saját komponensünket. Még olyankor is érdemes ezt megtennünk, ha nem akarjuk újra használni valamelyik alkalmazásspecifikus formunkat, csak könnyen elő szeretnénk varázsolni munkánk folytatásához például egy legutóbbi alkalmazás formját.

Gondolatmenetét Scott még egy figyelemre méltó ötletnek a felvetésével fejezi be. A Delphi tervezési interfésze arra is felhasználható, hogy létrehozzunk egy komponensszerkesztőt, és ennek segítségével tervezés közben teszteljük a formkomponenseket. Ha az objektumvizsgáló tulajdonságait betöltjük, és duplán kattintunk a komponensre, akkor ott a form mindjárt megjelenik, anélkül, hogy az egész alkalmazást le kellene fordítanunk. Így a „mi lenne, ha...” típusú gyors teszteket akár anélkül is ki lehet próbálni, hogy egyetlen sor kódot kellene írnunk.

Befejezésül csak annyit, hogy aki több információt szeretne kapni a felvetett ötletek megvalósításáról, nyugodtan bombázza további részletekért a Borland magyarországi képviseletét.

Vargha Dénes

OOP és OOT

Sokszínűség

Az OOT módszerek leginkább jelöléseikben (notációjukban) térnek el egymástól, nem pedig alapelveikben. Sokakat megtéveszt a jelölések sokszínűsége, és azt hiszik — sőt vannak, akik hangoztatják is —, hogy az OOT módszertanok még nem elég kiforrottak, nem elég egységesek. Pedig...

Az objektumorientált tervezés (OOT) olyan szoftverkészítést támogató módszer (módszertan), amely koncepcióinak nagy részét az objektumorientált programozásból (OOP) vette át. Bármelyik OO rendszert vizsgáljuk is meg, mindegyik a következő koncepciók együttesével jellemezhető: identitás, klasszifikáció, bezárás, öröklés, többalakúság. Vegyük sorra ezeket az alapfogalmakat.

Az identitás (identity) azt jelenti, hogy a szoftver (szoftverrendszer) adatai és eljárásai (viselkedése, szolgáltatásai) diszkrét, pusztán a létük által elkülöníthető, objektumoknak nevezett elemekbe szerveződnek. Az objektumokat megváltoztathatatlan egyedi azonosítóval látja el a rendszer. Az objektumazonosító szerves része az objektumnak, nem egyszerűen csak (a rendszertervező által kitalált) attribútuma. Két objektum akkor is külön objektumnak tekintendő, ha minden attribútuma azonos.

Fő fogalom: az osztály

A klasszifikáció során az objektumokat attribútumaik és viselkedésük azonossága vagy hasonlósága alapján osztályokba csoportosítják. Minden objektum beletartozik egy (vagy több) osztályba, és az objektum tudja is, hogy melyik osztály(ok)ba tartozik. A tervezés során az osztályokat attribútumaikkal és műveleteikkel definiáljuk. Az osztály az OOT legfontosabb fogalma. Az OOT alkalmazása azt jelenti, hogy osztályokat tervezünk: az OOT minden tevékenysége arra irányul, hogy elkészítse a megálmodott szoftver (implementációs) osztályainak specifikációját, beleértve ezek attribútumainak és műveleteinek a specifikációját is. A programozók és a tervezőeszköz kódgenerátorai azután ezeket az osztályokat alakítják ki az adott programozási környezetben.

A bezárás az osztályoknak az a képessége, hogy szabályozni tudják attribútumaik és műveleteik hozzáférését. A hozzáférés szabályozása közelebbről annyit jelent, hogy azok az attribútumok és műveletek, amelyek csupán arra valók, hogy realizálják az osztály objektumainak publikus szolgáltatásait, a külvilág számára nem érhetők el (privátak, védettek). Legkifinomultabb formájában a bezárás az interfészek közvetlen alkalmazásában nyilvánul meg. Az interfész olyan (eredendően) publikus műveletek együttese, amelyekkel leírható egy jól meghatározott szolgáltatás (contract). Az interfészeket osztályok valósítják meg. A lehető legjobban integrálható rendszereket interfészek következetes alkalmazásával lehet készíteni.

Az öröklés az osztályoknak azon képessége, hogy származási hierarchiába tudnak szerveződni, azaz egyik osztály származhat más osztályokból. Vannak olyan osztályok, amelyeknek nincsenek is objektumaik (ezek az ún. absztrakt osztályok). Ezeket csupán azért hozzák létre a tervezők, hogy hatékonyabban implementálhassák a rendszert.

A többalakúság azt jelenti, hogy másmás osztályok objektumain ugyanaz a művelet eltérően viselkedhet. Az öröklési hierarchiában ugyanolyan nevű és ugyanúgy alkalmazható műveleteket definiálhatnak az osztályok, mint őseik, azaz felüldefiniálhatják őseik egyes műveleteit. Egy adott helyzetben a futtatórendszer (Java virtuális gép, C++ runtime) a tényleges objektumok típusa (osztálya) alapján dönti el, hogy melyik műveletet kell végrehajtani.

Segítő módszertanok

Minden OOT módszertan (OMT, Booch, Coad, Jacobson, Kiss, Fusion stb.) alapvetően a fenti elvek szerint működik. Ha valaki egy OOT módszert megtanul, akkor nagyon könnyen eligazodik a többiben is. Jelölésbeli különbségeik ellenére nagyon is egységes ez a világ, hiszen mindegyikük ugyanazokkal az alapfogalmakkal dolgozik.

Az ábrázolás sokféleségében való eligazodást is nagyban elősegítette az UML, a Unified Modeling Language megjelenése. Az UML egységes OOT jelölésrendszer, amelyet az Object Management Group (OMG) már szabványként is elfogadott. A Rational cégnél alakították ki olyan jó nevű főtechnológusok (J. Rumbaugh, G. Booch, I. Jacobson), akik korábban maguk is önálló módszereket fejlesztettek ki.

Igaz, hogy máig sem minden OO módszertan kidolgozója („főguruja") fogadja el az UML jelöléseit, de ma már mindenki megadja modelljei UML ábrázolású változatát is. Az UML széles körű támogatottságot élvez a vezető szoftverházak körében is (IBM, Oracle, Microsoft), és várhatóan ez lesz az egyetlen elfogadott ábrázolási mód az OO világban. Sok szakember hangoztatja azt a véleményt, hogy használhatóságát és gyakorlatiasságát tekintve az UML gyengébb, mint a szerzők korábbi önálló alkotásai (egyesek szerint sokkal gyengébb) — de hiába, a kompromisszumoknak ára van. (Több is olvasható az UML-ről a [1] és [2] művekben.)

Elemzési módszerek

A legvitatottabb terület az OO technológián belül az OOT folyamatnak az elemzéssel foglalkozó része. Mivel a tervezési alapelvek az OOP-ből származnak, nem meglepő, hogy még az elemzés, ez a programozástól legtávolabb eső terület is élvezte az OOP megtermékenyítő hatását. A legpraktikusabb megközelítés sok szakember szerint (magamat is közéjük számítva) az OMT (1991) elemzési módszertan (l. [3] 8. fejezet). Érdemes megemlíteni, hogy az OMT 1991-es kiadásának még a jó öreg adatáramlási diagramokon alapuló funkcionális modellezés is szerves része volt. Én a magam részéről sajnálom, hogy az UML-ből ez kimaradt.

Ma már nem lehet olyan OO tervezőeszközt találni, amely támogatná az adatáramlási diagramokat. A funkcionális modell helyét átvette a Jacobsonféle felhasználási esetmodell (use case model), bár szerintem ez legalább annyira nem objektumorientált, mint az OMT'91 funkcionális modell, és semmivel sem hasznosabb, mint az adatáramlási diagramok voltak. Az OO elemzési technológia fejlődésére kétségtelenül I. Jacobson módszere gyakorolta a

legnagyobb hatást. A Jacobson által bevezetett felhasználási esetek (use case) és az azokból összeállított diagramok beépültek az UML-be, sőt ma már (Rumbaugh cikkei nyomán) maga az OMT is kiegészült a felhasználási esetek modellezésével. Talán nincs is már olyan OO tervezőeszköz, amely ne támogatná a felhasználási eseteket. (A részletek iránt érdeklődőknek a szakirodalmi jegyzékből két nagyon jó könyvet ajánlhatok figyelmébe: [1], [4].)

Az én gyakorlatias gondolkodásomhoz közel áll P. Coad módszere is ([5], [6]). Főleg azoknak ajánlom, akik nem szeretik a nagy hajcihővel, ceremóniával, terjedelmességgel járó elemzési eljárást, hanem kezdettől fogva a készítendő szoftver lebeg a szemük előtt. Jól használható (kiegészítő) módszer a CRC (Class, Responsibility, Collaboration) kártyák módszere is, amely — szemben a felhasználási eset- és funkcionális modellezéssel — teljesen objektumorientált. A CRC módszer a legforradalmibb elemzési technika (lásd [7]), éppen ezért viszont nem is ajánlom azoknak, akiknek a fejében túlságosan megkövesedett az egyedtípus-kapcsolat modellezési technika.

A fenti elemzési technikák, módszerek bizonyos kombinációkban vegyíthetők is, feltéve, hogy a tervezőeszközünk kellő támogatást biztosít hozzá.

Jó tervezőeszköz kell!

Az OOT megfelelő tervezőeszköz (CASE, Computer Aided Software Engineering) nélkül a gyakorlatban alkalmazhatatlan. A jó tervezőeszköz legfontosabb szempontjai a következők:

— Legyen megbízható a tervek és diagramok tárolásában.

— Legyen könnyen kezelhető és gyors.

— Támogassa a legfontosabb diagramfajták rajzolását.

— Közvetlen vagy közvetett módon támogassa az adattárban elhelyezett elemek (osztályok, metódusok, asszociációk stb.) metaattribútumokkal való bővítését.

— Tegye lehetővé az adattárra épülő generátorok írását (olyanokét is, amelyek magában az adattárban képeznek újabb elemeket).

— Támogassa az adattár szétosztását és a konfigurációkezelést (vagy legalábbis a konfigurációkezelőkkel való integrálást).

— Tegye lehetővé az adattár tartalmának a böngészését, azaz ne csak a diagramokon keresztül lehessen hozzájutni az adattár elemeihez.

— Támogassa a sorosított konkurens csoportmunkát.

— Ne legyen túl drága.

Nem soroltam fel a legfontosabb kritériumok között a kódgenerátorok meglétét, aminek az az oka, hogy én még nem láttam olyan CASE eszközt, amelynek a kód- vagy dokumentációgenerátora megfelelő lett volna. A C++ kódgenerátorok állnak talán a legközelebb ahhoz, hogy használhatónak lehessen minősíteni őket. Minden nagyobb fejlesztési környezet megkívánja, hogy a generátorokat projektjeink igényeihez igazítsuk. A legfontosabb az, hogy olyan kódgenerátoraink legyenek, amelyeket jól ismerünk, és egyszerűen át tudunk igényeinknek megfelelően alakítani.

Elterjedt szokás, hogy a CASE eszközök valamilyen Basic-szerű programnyelvet adnak a kódgenerátor írói számára. Jó kódgenerátort nem lehet Basic-szerű nyelven írni. Mivel a kódgenerátor a legfontosabb része egy nagyobb fejlesztési technológiának, maguk a kódgenerátorok is gondos tervezést és dokumentációt igényelnek. Szerencsére a tervezőeszközök lehetővé teszik, hogy általános módon is hozzáférjünk az adattárhoz (például ActiveX felületen, szövegfájlként, ObjectStore API-n keresztül stb.), és így normális programnyelven (C++, Java, Perl, Awk, Tcl/Tk stb.) írhatjuk a generátorokat.



Adatbázisok tervezése

Az OO eleinte csak eszközként használta az adatbázisok technológiáját, de ma már úgy látszik, nagyobb rendszerek esetében át is veszi az irányítást az adatbázisok fejlesztésében.

A relációs adatbázisok tervezése az adott adatbázisban alkalmazott elemek tervezését jelenti (táblatér, tábla, index, nézet, tárolt procedúra, szerep stb.). Mivel nincs két azonosan definiálandó adatbáziskezelő, a részletes adatbázistervezési technológiát adatbáziskezelő-specifikusan lehet csak létrehozni.

A végső cél az adatbázist előállító script generálása. A scriptet futtató eszköztől függ, hogy milyen legyen maga a script (például egy Oracle SQL*Plus számára készített scriptet egy Microsoft-eszköz nem tudja lefuttatni).

Az alkalmazandó adatbázistervezési technológiát is nagymértékben meghatározzák a tervezőeszköz szolgáltatásai. A Platinum Paradigm Plus például közvetlenül támogatja a relációs adatbázistervezést (ERM diagramokat), míg a Rational Rose-ban az osztálydiagramokat lehet jól felhasználni az adatbázis megtervezéséhez.

Nincs olyan relációs adatbáziselem, amelyet ne lehetne megtervezni akármelyik OO tervezőeszközzel. Legrosszabb esetben magunk módosíthatjuk a kódgenerátort, vagy akár teljesen újat írhatunk saját ízlésünk szerint. Jó néhány olyan hazai céget ismerek, amelyek saját céljaikra, saját elgondolások szerint készítettek vagy készíttettek adatbázis-generátort az általuk használt OO tervezőeszközhöz.

A nem relációs OO adatbázisok tervezése tulajdonképpen sokkal egyszerűbb, mint a relációsaké: a feladat szinte teljesen azonos a C++ vagy Java programok tervezésével. Nagy segítséget jelent, hogy az OO tervezőeszközökhöz igen jó minőségű C++ és Java kódgenerátorok állnak rendelkezésre, sőt a legelterjedtebb OO CASE eszközökhöz a legtöbb ODBMS-t gyártó cég ingyen letölthető generátorokat is készít.

Németh Miklós

SZAKIRODALOM

[1] M. Fowler, K. Scott: UML Distilled. Applying the Standard Object Modeling Language. (Reading, MS: Addison Wesley Longman), 1997

[2] H. Eriksson, M. Penker: UML Toolkit. (New York, NY: John Wiley & Sons), 1998

[3] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, F. Lorensen: Object-Oriented Modeling and Design. (Englewood Cliffs, NJ: Prentice-Hall), 1991

[4] I. Jacobson, M. Ericsson, A. Jacobson: The Object Advantage. Business Process Reengineering with Object Technology. (Reading, MS: Addison Wesley Longman), 1995

[5] P. Coad, M. Mayfield: Java Design. Building Better Apps and Applets (Englewood Cliffs, NJ: Prentice-Hall, Yourdon Press), 1997

[6] P. Coad, D. North, M. Mayfield: Object Models: Strategies, Patterns and Applications. (Englewood Cliffs, NJ: Prentice-Hall, Yourdon Press), 1995

[7] N. Wilkinson: Using CRC Cards. An Informal Approach to Object-Oriented Development. (New York, NY: Sigs Books), 1995

A modellkészítés nem kerülhető el

Álom és valóság

Megfelelő tervezőeszköz birtokában és megfelelelő tudással felvértezve hozzáláthatunk megálmodott rendszerünk megtervezéséhez. Nézzük meg most folyamatában, mi kell ahhoz, hogy elkészülhessen a rendszer.

Bármilyen elemzési módszert használunk is, mindenképpen el kell készítenünk a rendszer elemzési modelljét. Ennek a modellnek a részletes rendszertervezés számára legfontosabb részei az üzleti objektummodell (business object model) és a felhasználói felület navigációs modellje. (Ez még kevesebb, mint a részletes felhasználói felület modellje, de annak legfontosabb elemeit már tartalmazza.)

Az üzleti objektummodell a strukturált módszertanok fogalmi adatmodelljére hasonlít, azzal az eltéréssel, hogy az üzleti objektummodell tartalmazhat tranziens (átmeneti, adatbázisban nem tárolt) osztályokat is. Pontosabban: az üzleti objektummodell a rendszert alkotó üzleti fogalmakat reprezentáló, attribútumokkal ellátott osztályokat és kapcsolataikat tartalmazó modell, amely az osztályok leírásaiból és osztálydiagramokból áll.

Üzleti objektumok tervezése

A háromrétegű (three tiered) rendszerek három rétege a megjelenítésért felelős prezentációs réteg, az alkalmazásszerver és az adatbázis. Ezek közül a tervezés legfontosabb része a szerverkomponens (vagy más néven üzleti objektum) tervezése. Ez a tervezés Corba (Common Object Request Broker Architecture), Microsoft DCOM (Distributed Component Object Model) vagy Java RMI (Remote Method Invocation, távoli eljáráshívás) interfészek definiálását jelenti. Ha a fejlesztői stábot külön kliens- és szerveroldalt fejlesztő csapatra osztjuk, akkor feltétlenül definiáljuk minél előbb a szerverkomponenseket és azok interfészeit. Ha ezt sikerül megtennünk, akkor a két csapat már egymástól függetlenül tud dolgozni.

A szerverkomponensek szolgáltatásainak megtervezéséhez először is az kell, hogy világosan meg legyenek fogalmazva a rendszerrel szemben támasztott követelmények, például felhasználói esetek és forgatókönyvek formájában. Ehhez tulajdonképpen nem szükséges a tényleges felhasználói felület ismerete. A megrendelő azonban a legtöbb esetben csak úgy tudja megítélni, hogy megfelel-e elképzeléseinek a rendszerterv, ha a felhasználói felület szolgáltatásain keresztül szemléletesen követni tudja, hogy mikor mi történik. Ezért fontos, hogy a felhasználói felület navigációs modelljét a fejlesztési folyamat korai fázisában elkészítsük, mert a szerverkomponensek tervezéséhez is az abban megfogalmazott feladatok szolgálhatnak alapul.

Kapcsolat a felhasználóval

A rendszer elemzési modelljének igen fontos része a felhasználói felület navigációs modellje. A megrendelőknek saját jól felfogott érdekükben nem is szabad elfogadniuk olyan modellt, amelynek elemzési fázisában nem készülnek el a felhasználói felületre vonatkozó tervek. Hogy mit kell tartalmaznia ennek a navigációs modellnek? A felhasználói felületet reprezentáló osztályokat, ezek navigációs kapcsolatait és a képernyőterveket.

Ennyi már elég ahhoz, hogy az elemzési modellből megfelelő tervezőeszközzel és generátorokkal előállítható (generálható) legyen a felhasználói felület osztályainak nagy része. Az iteratív fejlesztés annyival bonyolultabb, hogy ennél egyszerre csak egy rendszerrész (ún. partíció) modellje készül el. Az iteráció folyamatában a partíciókra való tagolást szakterületi szempontok szerint kell meghatározni, annak megfelelően, hogy éppen melyik a megtervezendő terület (például ügyfélkezelés, számlázás, beszerzés stb.). Ezt a partíciókra bontást szoktuk „függőleges tagolásnak" tekinteni.

Itt jegyezzük meg, hogy ha más rendszerekhez illeszkedő alkalmazásszervert kell létrehoznunk, akkor az üzleti objektummodell helyett üzleti komponensmodellt kell készíteni, amely a szerver szolgáltatásait definiálja (Corba IDL-ek).

Tervezés és kódgenerálás

Az elemzési fázis végén keletkezett modelleket a megrendelővel jóvá kell hagyatni, csak ezután következhet a rendszer (az osztályok) részletes tervezése. Az üzleti objektummodellből kiindulva készíthetők el a rendszer technikai szempontból („vízszintesen") tagolt részei (rétegek, layers): a relációs vagy objektumorientált adatbázismodell, a szerveroldali komponensmodell (Corba IDL-ek és implementációs osztályok), végül a részletes felhasználói felület és a vezérlési modell.

Ha megfelelő tervezőeszközzel és generátorokkal rendelkezünk, akkor az üzleti objektummodell alapján generálni lehet ezeket a rendszerrészeket. A tervezők azután tovább finomíthatják az egyes elemeket. Végül a fokozatos finomítás eredményeképpen a kódgenerátorok már olyan minőségű programkódot gyártanak a modellrészekből,



— Most már erre az objektumra orientálódom.

hogy kézi módszerekkel már csak a speciális műveletek (metódusok) testét kell beprogramozni.

Mennyire legyen részletes a tervezőeszközben tárolt rendszerterv? Ennek eldöntésénél az az alapelv, hogy a programozónak már ne kelljen új osztályt és publikus metódust definiálnia. Szigorúbb követelményt is felállíthatunk: még privát tagokat se vehessen fel a programozó. A szoftver teljes struktúrája a részletes modellekből generálódik. A mai generátorok olyanok, hogy a forráskódnak a programozó által írt részeit egy az egyben megőrzik. Azokat a részeket, amelyeket a tervezőeszköz generál, a programozó már nem módosíthatja.

Teljes felületterv

A felhasználói felület tervezése a képernyőosztályok, a képernyőosztályokat vezérlő osztályok és a szerverkomponensekkel kapcsolatot tartó osztályok (ún. lokális üzleti objektumok vagy proxyk) tervezéséből áll.

A proxyk tervezésére normál esetben nem is kell külön erőfeszítéseket tenni, mivel azok automatikusan előállnak a szerverkomponensek tervezésével. Csak az olyan proxyk tervezését kell külön elvégezni, amelyek speciális szolgáltatásokat vannak hivatva elvégezni.

A felhasználói felület objektumainak realizálásához felhasználhatjuk a legkülönbözőbb technológiákat, webalkalmazás esetén a képernyőobjektumokat elkészíthetjük html, szimpla Java AWT, Borland JBCL, Symantec Visual Café, Marimba Bongo, Netscape IFC, SunSoft JFC, Microsoft ActiveX, KLGroup JClass stb. elemekből. Ilyen bőség láttán arra kell vigyázni, hogy a képernyőobjektumok ne tartalmazzanak (szó szerint ne tartalmazzanak) semmiféle „vezérlési okosságot”. Egy rendszer adatbáziskezelője ugyanis nem szokott megváltozni, a felhasználói felület viszont többször is megváltozhat, sőt egyszerre többféle felhasználói felület, technológia is előfordulhat egy rendszerben.

Németh Miklós

SZAKIRODALOM

[1] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, F. Lorensen: Object-Oriented Modeling and Design. (Englewood Cliffs, NJ: Prentice-Hall), 1991

[2] M. Fowler, K. Scott: UML Distilled. Applying the Standard Object Modeling Language. (Reading, MS: Addison Wesley Longman), 1997

[3] P. Coad, M. Mayfield: Java Design. Building Better Apps and Applets (Englewood Cliffs, NJ: Prentice-Hall, Yourdon Press), 1997

[4] P. Niemeyer, J. Peck: Exploring Java. Second Edition. (Cambridge: O'Reilly), 1997

Előny a fejlesztésnél

OOP adatbázisok

Nagy rendszerek fejlesztésében — különösen, ha azok egész életpályáját, változásait is figyelembe vesszük — rendkívül markánsan megmutatkoznak az OOP előnyei. Ma már úgy látszik, hogy az itt alkalmazott módszerek az egész adatbázis-technológiát forradalmasítják.

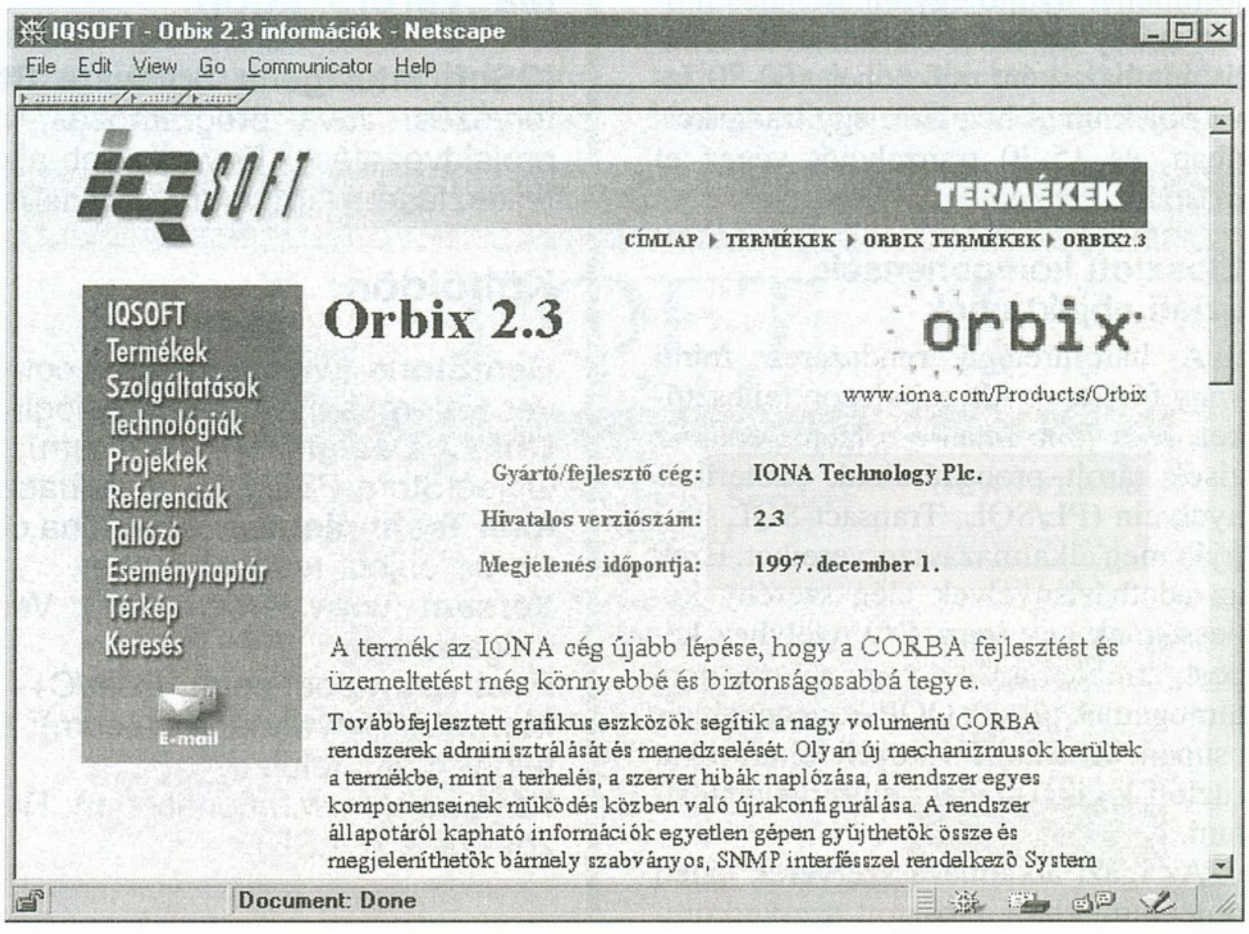
Azok a C++, illetve Java objektumok, amelyekkel a programozó a memóriában dolgozik, ideálisak arra, hogy úgy, ahogy vannak, adatbázisba kerüljenek. A feldolgozó program induláskor az adatbázisból tölti be az objektumokat a memóriába. Az objektumok kapcsolataik mentén tetszőleges méretben és bonyolultságban csoportosíthatók (clusterezhetők), és ezek a csoportok hatékonyan mozgathatók.

Egy objektum-adatbázis akkor hatékony, ha nagyon sokfajta objektum bonyolultan van összekapcsolva. Olyan rendszerek számára, ahol bonyolult kapcsolatokat kell kezelni nagy sebességgel, az objektum-adatbázisok verhetetlenek. Ilyenek a CASE és CAD eszközök adattárai (repository), különböző dokumentumtárak (sgml, html, xml, multimédia), telekommunikációs cégek adatbázisai stb. Nagy adatmennyiség esetén a relációs adatbáziskezelők képtelenek elfogadható sebességgel összekapcsolni (join) négy vagy annál több táblát — hasonló művelet egy objektum-adatbázisnak meg sem kottyan. Akár 50-100 kapcsolatot is simán tud kezelni.

Adatbázisok versengése

Méréseink szerint az ODBMS-ek másodpercenként 1000-3000 objektumot képesek létrehozni egyetlen tranzakcióban, ezzel szemben egy jó nevű cég egyébként kiváló relációs adatbázisszervere másodpercenként alig 80-100 sort.

Akkor hát az objektum-adatbázisok miért nem szorítják ki a relációs adatbázisokat? A folyamat már megkezdődött, de lassan halad, ugyanis olyan hatalmas méretű tapasztalat és tudás gyülemlett fel a relációs (SQL) adatbázisok alkalmazásainak kifejlesztésében, hogy ez csak lassan engedi át a helyét az új technológiáknak. (Így volt ez korábban is, a Cobol adatbázisokkal kapcsolatban.) A nagyvállalatok nem tudnak meglenni adatbázisok nélkül.



Ha valahol működik egy adatbázis, azt csak akkor cserélik le, ha az új alkalmazás sokkal jobb szolgáltatásokat nyújt. Az objektum-adatbázisok terjeszkedésére csak a teljesen új rendszerek létesítésekor van igazán lehetőség. Jellemző, hogy a webes alkalmazásokban jóval magasabb az objektum-adatbázisok aránya, mint a hagyományos rendszerekben.

SQL és OQL

Az objektum-adatbázisok valójában csak C++ programozók számára jelenthetnek alternatívát. A C++-műhelyek számára azonban az objektum-adatbázisok használata sokkal termelékenyebb, mint SQL szervert alkalmazni, főleg akkor, ha az objektum-adatbázis az SQL-nél is sokkal több szolgáltatást nyújtó OQL (Object Query Language) nyelven le is kérdezhető (Versant, Poet). A C++-programozók számára az objektum(adat)modell a memóriában van.

Nagy erőkkel folyik a Java nyelv illesztése a meglévő C++ adatbázisokhoz. Méréseink szerint azonban jelenleg még a JDBC-vel meghajtott SQL szerverek általában jobb teljesítményt adnak, mint a C++ objektum-adatbázisok Java interfészei. Igazi megoldás az lehet, ha már kifejezetten a Javához készítenek adatbáziskezelőket; ilyenen dolgozik például a GemStone és az ODI. A GemStone/J egy tranzakcióban 600 Java objektumot képes másodpercenként létrehozni, de gondok vannak még a stabilitással és a tranzakciókezelési sebességgel. Az ObjectStore PSE Pro for Java teljesen Javában írt tranzakciókezelést biztosító, megfelelő teljesítményt nyújtó egyfelhasználós motor, amely Castanet csatornákban lokális adatbázisként működhet. (60-80 Java objektumot hoz létre egy tranzakcióban, és 15-30 tranzakciót végez el másodpercenként.)

Elosztott komponensek, üzleti objektumok

A háromrétegű rendszerek iránti vágy fűtötte és fűti mindazon fejlesztőket, akik jobb híján a relációs adatbázisok tárolt procedúráinak ezoterikus nyelvein (PL/SQL, Transact-SQL stb.) írják meg alkalmazásszervereiket. Ezek az adatbázisnyelvek elég szerény képességűek egy igazi OO nyelvhez képest, amilyen a C++ vagy a Java, nem támogatnak túl sok OOP-koncepciót, és csupán az adatbázisközeli eljárásokat (üzleti logikát) lehet velük megvalósítani.

Az igazi alkalmazásszerverek üzleti objektumokba („elosztott szerverkomponensekbe") szervezett interfészeket bocsátanak a felhasználók (kliensprogramok) rendelkezésére. Ezek az üzleti objektum interfészek valamelyik szabványos elosztott technológiával érhetők el (lehet ez Corba, Microsoft DCOM vagy Java RMI), függetlenül attól, hogy milyen adatbáziskezelő áll mögöttük, és hogy milyen nyelven írták őket (C, C++, Java, Visual Basic 5 — ez utóbbival csak ActiveX DCOM objektumokat lehet készíteni). Feladataik adatbázisközeli részeinek megoldására maguk a szerverobjektumok természetesen behívhatnak adatbázisban tárolt procedúrákat.

A kliensoldalon ezeknek a szerveroldali üzleti objektumoknak a proxy objektumok felelnek meg. A proxy objektumok sok olyan feladatot is elláthatnak, amelyekhez nem kell a szerverhez fordulni: végezhetnek adatellenőrzéseket, kezelhetnek kódtáblákat, szolgáltathatnak az üzleti objektumokra vonatkozó metainformációkat stb.

Felhasználói felületek

A felhasználói felület a lehető legkülönfélébb módon valósulhat meg ugyanazon alkalmazáshoz: (dinamikus) html formok (JavaScriptekkel), Java AWT, Marimba Bongo, Microsoft ActiveX, Netscape IFC stb. Hozzájuk képest az adatbázis és az üzleti logika sokkal kevésbé változékony elemei a rendszernek. Az alkalmazás üzleti logikájának Corba interfészeken keresztül elérhető üzleti objektumokba (szerverkomponensekbe) való tudatos szervezése biztosítja a fejlesztőknek, hogy rendszereik sokkal időtállóbbak lesznek, továbbá később is integrálhatóak maradnak, ugyanis a változások csupán a lehető legkisebb rendszerrészre (ideális esetben egyetlen komponensre) korlátozódnak.

Egyes 4GL-ek (Forté, Composer, PowerBuilder, Supernova) szintén képesek elosztott objektumok kifejlesztésére és kezelésére, de ezek (egyelőre legalábbis) csak az adott eszköz környezetéből érhetők el, és (egyelőre) nem kompatibilisak a szabványos elosztott middleware technológiákkal (Corba, Java RMI, DCOM).

Németh Miklós

SZAKIRODALOM

[1] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, F. Lorensen: Object-Oriented Modeling and Design. (Englewood Cliffs, NJ: Prentice-Hall), 1991

[2] M. Fowler, K. Scott: UML Distilled. Applying the Standard Object Modeling Language. (Reading, MS: Addison Wesley Longman), 1997

[3] P. Coad, M. Mayfield: Java Design. Building Better Apps and Applets (Englewood Cliffs, NJ: Prentice-Hall, Yourdon Press), 1997

[4] P. Niemeyer, J. Peck: Exploring Java. Second Edition. (Cambridge: O'Reilly), 1997

Az OO technológia elkötelezettjei

Magyarországon:

IQSoft Intelligens Software Rt (www.iqsoft.hu): Konzultáció (OO tervezés, Java programozás, relációs és OO adatbáziskezelés, projektvezetés). Egyedi web-alkalmazások (Java, html, Castanet) fejlesztése. Hagyományos kliens-szerver alkalmazások fejlesztése.

Külföldön:

GemStone (www.gemstone.com): GemStone/J Java adatbázisszerver EnterpriseBeans technológiával.
Object Design (www.odi.com): ObjectStore/C++ adatbázisszerver, ObjectStore PSE Pro egyfelhasználós Java adatbázismotor.
Iona Technologies (www.iona.com): Orbix/C++ és OrbixWeb (Java) Corba object request broker.
Versant (www.versant.com): Versant/C++ adatbáziskezelő OQL támogatással.
Poet (www.poet.com): Poet/C++ adatbáziskezelő.
Marimba (www.marimba.com): Castanet webtechnológia és Bongo felhasználói felület.
Rational (www.rational.com): Rational Rose objektumorientált tervezőeszköz (CASE).

Átalakuló adatbázisok

Jobbak, mint a relációsak

Gyakran hetek alatt kell elkészíteni nagyméretű, összetett alkalmazásokat az Internet gyorsan változó és növekvő szolgáltatási igényeinek kielégítésére. Az eszköz, amellyel a cégek eleget tudnak tenni ennek az óriási kihívásnak, az objektumorientált technológia és a komponensalapú fejlesztés.

Az OOT és OOP azokon a területeken is lendületesen terjeszkedik, amelyeket hagyományosan a relációs adatbáziskezelő rendszerek uraltak. Az ok nyilvánvaló: az OOP környezetekbe természetesebben illeszkednek be az OO adatbázisok. Számos működő alkalmazás bizonyítja, hogy az OO adatbáziskezelők nemcsak el tudják látni mindazokat a szolgáltatásokat, amelyeket a relációs adatbáziskezelők, hanem megbízhatóságban és különleges követelményeket támasztó alkalmazások kivitelezésében felül is múlják őket.

Többet és jobban

Az OOAK (objektumorientált adatbáziskezelő) gyakorlati megjelenése egybeesett a CAD, CAM, CASE stb. eszközök megerősödésével, terjedésével. Eleinte ezeket az alkalmazásokat általában C++ nyelven készítették el — nagy bonyolultságuk, illetve az alkalmazott adatszerkezetek sokfélesége és összetettsége miatt. Éppen az ilyen feladatok megoldására kiválóan alkalmasak az OOAK rendszerek, mert roppant egyszerűen tudnak tárolni és visszatölteni nagyszámú bonyolult adatszerkezetet, ráadásul a C++ nyelvben megírt adatkezelő programokhoz is igen jól illeszkednek. Kezdetben még előfordult, hogy a Smalltalk nyelvet használták az OOAK elérésére. Ma ennek a nyelvnek a szerepe már csökken, helyét a dinamikusan terjedő Java veszi át.

Egy OOAK úgy is felfogható, mint egy adott programozási nyelv (C++, Java) természetes kiterjesztése. A kiterjesztést az indokolja, hogy bizonyos feladatok megoldásához különleges eszközöket kell igénybe venni az objektumok ún. perzisztens tárolásának biztosítására. (A perzisztencia szó jelentése: megmaradás, szívósság, állandóság, folytonosság. A persistency szóval fejezik ki az angolban például az energia megmaradását is.) Az OOAK egyik nagy előnye a relációs rendszerekkel szemben, hogy ugyanolyan formában tárolják az objektumokat, ahogy azok a memóriában megtalálhatók. Nincs tehát szükség arra, hogy az adatbázisba való elhelyezés előtt, illetve az adatbázisból való kiolvasás után átalakítsuk az objektumokat. A programozók számára egy perzisztens objektum pontosan olyan, és ugyanúgy használható, mint egy memóriában tartott, ún. tranziens objektum. (Csak mellékesen említjük meg, hogy az objektumok sorában lehetőség van ActiveX objektumok elhelyezésére is.)

Navigáció közelre és távolra

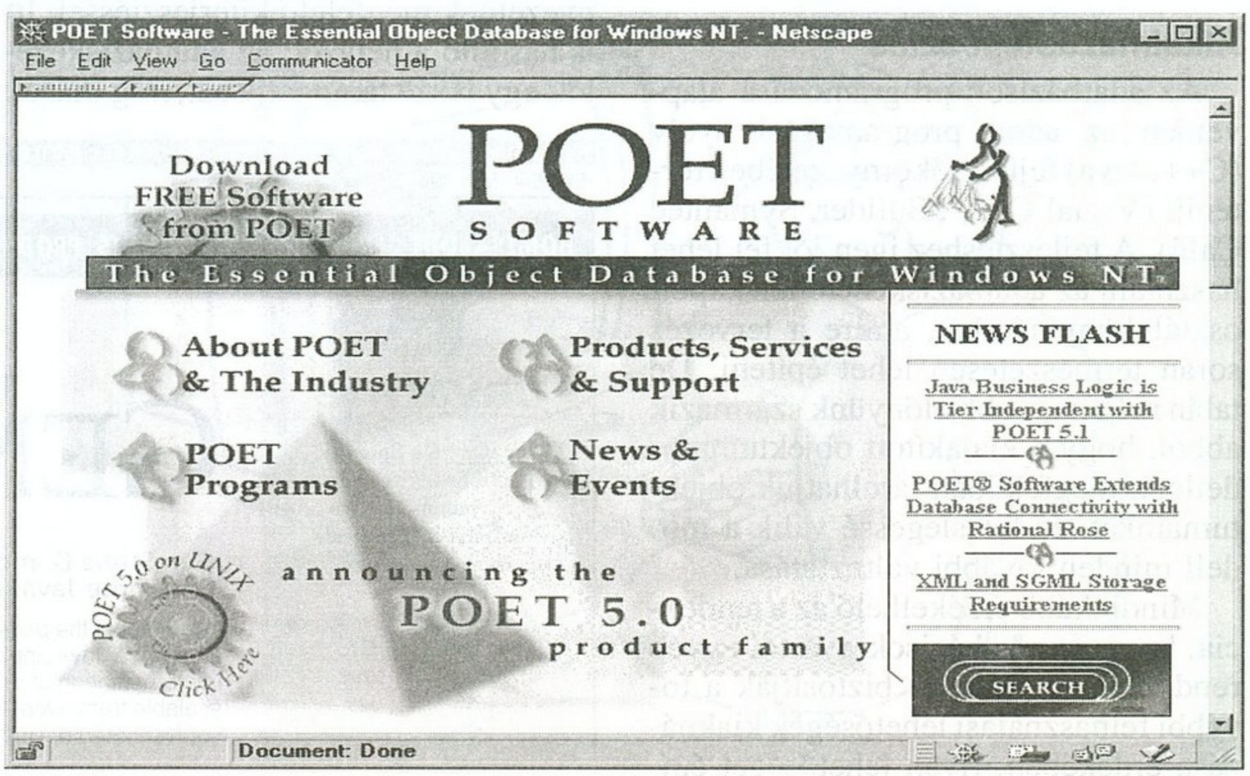
Az objektumok közötti kapcsolatokat, mozgásokat mutatók (pointerek) teremtik meg. Ezt a tevékenységet nevezik az OOAK zsargonjában navigációnak. A navigáció igen gyors adatelérést tesz lehetővé. Ha a kapcsolódó objektum már a memóriában van, akkor pillanatok alatt elérhető az adat. De ha a hivatkozott objektum még nincs a memóriában, akkor sincs szükség keresésre, elég a háttértárról való betöltés idejét figyelembe venni. Ez ugyancsak hatalmas előny a relációs adatbáziskezelőkkel szemben. De más előnye is van a navigációnak: ennek révén az objektumok kapcsolatainak a tömege is könnyedén kezelhetővé válik.

Az OOAK biztosítja a relációsoknál már megszokott indexelési lehetőségeket is a lekérdezések hatékony végrehajtására. De többről is szó van. Az OOP nyelvekben kiváló lehetőségek vannak több objektum összefogására, ún. „kollekciók" (listák, tömbök, halmazok) kialakítására. OQL (Object Query Language) nyelven és az adatbáziskezelő lekérdezőnyelvén a különböző lekérdezéseket egy vagy több kollekción belüli objektum megtalálására is meg lehet fogalmazni, és persze a rendelkezésre álló eszközökkel rendkívül gyorsan végre is lehet hajtani.

Az OOAK ma már rendelkezik mindazokkal a szolgáltatásokkal (tranzakciókezelés, online mentés és visszaállítás, replikáció, fail-over), amelyeket a relációs adatbáziskezelőknél megszoktunk. Nagy adatbázisok esetén ilyenek nélkül nem is volna elképzelhető a megbízható alkalmazások kialakítása.

A rendszer felépítése

Az OOAK rendszerek kliens/szerver felépítésűek, szerver- és kliensfolyamatokból állnak. Az objektumok kezelése bizonyos fajta tevékenységek végrehaj-



tásához teljesen a kliens hatáskörébe tartozhat (ilyenek a memóriába való betöltés és a különböző fajta lekérdezés); van úgy, hogy a szerver „nem is tud" az objektumokról. Más esetekben a szerver közreműködése szükséges az objektumokkal végzendő műveletek, lekérdezések végrehajtásához.

Az objektumok elérésének hatékonyságát növeli az objektumok csoportosítási (segment, cluster) lehetősége is. Ez a technika biztosítja, hogy fizikailag is „egymáshoz közel" helyezzünk el az adatbázisban logikailag összetartozó objektumokat, az össze nem tartozókat pedig „egymástól távol", ezzel is csökkentve az objektumok betöltéséhez szükséges lemezmozgást.

A kliensoldalon ún. objektum-cache szolgálja az OOAK teljesítményének növelését. Ez átmenetileg tárolja az egyszer már letöltött objektumokat, és a következő hivatkozásnál felgyorsítja az objektum elérését.

Mérési eredmények

Saját méréseink is igazolják, hogy OOAK· révén összehasonlíthatatlanul jobb eredményeket lehet elérni. Egy objektum és a hozzá kapcsolódó objektumok elérésében az objektumorientált rendszerek verhetetlenek a navigációs lehetőségnek, valamint az alkalmazott cache technikának köszönhetően. Ugyancsak kiváló teljesítményt nyújtanak, ha az egy tranzakción belül létrehozható objektumok számát nézzük: Windows NT alatt egy tranzakcióban másodpercenként 2000 körüli objektum létrehozása lehetséges (relációs adatbázisok esetén ez az érték 100 körül van). Érdekes viszont, hogy a másodpercenként végrehajtható tranzakciók számában valamivel elmarad a relációs adatbázisokban mért eredménytől.

Alkalmazásfejlesztés

Az adatbázisok programozása alapvetően az adott programozási nyelv (C++, Java) fejlesztőkörnyezetében történik (Visual C++, JBuilder, Symantec Café). A fejlesztéshez igen jól fel lehet használni az adatbáziskezelőhöz kapott osztálykönyvtárakat, amire a tervezés során természetesen lehet építeni. De talán még nagyobb előnyünk származik abból, hogy a kialakított objektummodellnek megfelelően tárolhatjuk objektumainkat, és feleslegessé válik a modell minden további változtatása.

Mindinkább érzékelhető az a tendencia, hogy az adatbázisok gyártói egyéb rendszerek elérését is biztosítják a további felhasználási lehetőségek kiaknázása érdekében. Ilyen lehetőséget építenek be például a webszerverek felé az SQL felület és az OLE DB felület támogatására (Oforms, ASP), vagy például interfészt teremtenek a különböző relációs adatbázisoknak.

Gyakori jelenség, hogy a cégek olyan egyszerű adatbázis-verziókat is készítenek, amelyek kisebb alkalmazások számára objektumtárakként működhetnek. Ezzel ugyanis egyrészt megnyerhetik a fejlesztők, programozók rokonszenvét, másrészt megkönnyíthetik az OOAK bevezetését. Némelyik ilyen egyszerű adatbáziskezelő ingyenesen is hozzáférhető.

Szabványosítási törekvések

Erőteljes szabványosítási törekvések figyelhetők meg az objektumorientált technológiák területén is. Az OOAK rendszereket fejlesztő cégek elsősorban szabványosítási céllal hozták létre az Object Data Management Group (ODMG) nevű szervezetet. Az ODMG 2.0 szabvány jól átgondolt előírásokat tartalmaz az objektummodellre. Ehhez csatlakozik az objektumleíró nyelv (ODL) szabványa, az objektumlekérdező nyelv (OQL) szabványa, de vannak már szabványok a nyelvi leképezésre vonatkozóan is, a C++, a Java és a Smalltalk nyelvre.

Az ODL segítségével az adatbázis szerkezetét is le lehet írni. Ez a nyelv valójában az Object Management Group (OMG) által már régebben specifikált interfészleíró nyelvnek, a Corba IDL-nek a kiterjesztése. Az adatbáziskezelők egy része már támogatja az adatbázisséma leírásában az ODL nyelvet, míg mások az adott programozási nyelven meglévő osztálydefiníciókat használják erre a célra.

Az OQL az SQL-hez hasonló lekérdezőnyelv, természetesen az OO környezetnek megfelelő kiterjesztéssel. Itt is hasonló a helyzet: az adatbáziskezelők egy része támogatja azt, míg mások a saját, többnyire a C++-hoz hasonló lekérdezőnyelvüket használják.

A nyelvi leképezés specifikációja mostanáig a C++, a Java és a Smalltalk programozási nyelvekhez készült el. A Smalltalk kivételével ezeket már minden gyártó támogatja, a Smalltalk nyelvi interfészét csak elvétve egy-egy.

Alkalmazási területek

Hagyományos alkalmazási területnek számítanak a CAD, CAM, CASE alkalmazások, de előszeretettel használják az OOAK rendszert minden olyan alkalmazásban, ahol folyamatosan nagy mennyiségű adatot kell az adatbázisba tölteni. Tipikusan ilyen alkalmazási terület a távközlésben a rendszerfelügyelet, vagy pénzügyi területen a kockázatelemző és a tőzsdei rendszerek.

A legdinamikusabban fejlődő területet az internetes adatbázisok jelentik. Itt jól kihasználható az objektumorientált adatbázisoknak az a páratlanul előnyös tulajdonsága, hogy az adatok mennyiségétől és az adatszerkezetek bonyolultságától függetlenül lehetővé teszik bármilyen jellegű adathalmaz (szöveg, kép, hang, videó, idősor stb.) hatékony tárolását és visszakeresését. Elsősorban az Internet az a terep, ahol az OO adatbázisok további előretörése várható. Erről tanúskodnak az adatbázisfejlesztő cégek újabb referenciái is, amelyek nagy többségükben erről a területről kerülnek ki.

Az OO adatbázisok terjedése olyan területeken várható igazán, ahol új adatbázisok kialakítása folyik. A meglévő adatbázisok átállítása nagy pénzeket emészt fel, és nem is biztos, hogy mindig célszerű. Átszervezésre főleg akkor érdemes gondolni, ha a rendszer nem nyújtja azt a teljesítményt vagy hatékonyságot, amit elvárhatnánk tőle.

Nick János
nick@iqsoft.hu



Lépcsőházi effektus

Szubjektumorientáltan

Az objektumorientált programozáshoz legkönnyebben hozzáférhető eszköz a Turbo Pascal, a Borland Pascal és a Delphi. Érdekes probléma, hogy milyen benyomások keletkeznek egy tapasztalt programozóban, amikor először próbálkozik meg az egekig magasztalt módszer gyakorlati alkalmazásával...

Tíz évvel ezelőtt gyakorlott Pascal-programozó — helyesbítek: Turbo Pascal-programozó — voltam. Ötödik éve használtam a Borland fejlesztőeszközét, kezdve a 3-as verzióval, amit még annak idején hálózatkezelési okokból meg is „pecseltem", békésen ücsörögve az 5-ös változat kényelmes fejlesztőkörnyezetében. Ez a rendszer mindent tudott, amire szükségem volt. Tudtam eljárásokat és függvényeket átadni paraméterként, ezekre könnyedén hivatkozhattam pointerekkel, így nem okozott gondot akár a bonyolult adatbeviteli képernyőket generáló programok megírása sem.

Ekkortájt is sokat olvastam a szaksajtót (nem újságíróként, kritikusként, hanem mert — nem lévén hozzáférhető Internet — információkhoz akartam jutni), és feltűnt, hogy egyre többet cikkeznek az ún. OOP módszerről. Szokva voltam az időnként mindent elborító divathullámokhoz, de ez meglepett. Egyszerűen nem értettem, hogy mire való, mi ez a nagy felhajtás.

Az OOP alapköve, hogy az adat és az azt kezelő eljárás egyben, egységben van. Ez csak szemléleti kérdésnek tűnt, programjaim szépen unitokra voltak bontva, elején az adat- és függvénydeklarációkkal, könnyen átlátható, hogy mi mihez tartozik — vitatkozgattam magamban. Engem mint programozót ugyan mennyiben érint ez?

Az objektumoknak az akkori terminológiával „üzeneteket" küldtünk, azok pedig „reagáltak" erre. Ez nagyon antropomorfnak látszik, de az objektumaimnak én nem üzengetni akarok! Programom eljárások sorozata, amelyeket az végrehajt.

Az objektumorientáltság mellett szólt, hogy egy kiinduló objektumból újakat lehet származtatni, ezek öröklik a szülő(k) „tulajdonságait", és rendelkeznek sajátokkal is. A felhozott példák rendszerint triviálisak voltak, jellemző volt a pont mint objektum (van x és y koordinátája, színe), majd kör, négyzet stb. mint származtatott objektum. Ennél értelmesebb példát sehol sem láttam.

Szintén az új eszme előnyeihez sorolták, hogy mások által készített objektumokat is használhatok, sőt azokat bővíthetem a már említett öröklődés segítségével. Ez sem nagyon izgatott, mindig egyedül dolgoztam, és amit mások készítettek, én mégis használtam, az valamilyen rutinkönyvtár volt (forráskóddal).

Az objektumokat ráadásul létre kell hozni, célszerűen futásidőben, azaz dinamikus memóriakezelésre van szükség, az objektumokat illik el is engedni (konstruktor és destruktor), és mindez teljesen rendezetlenül zajlik. Akkor pedig tele lesz lyukakkal az (akkoriban még szűkös) memória, a beépített takarórutinokban meg nem nagyon bíztam.

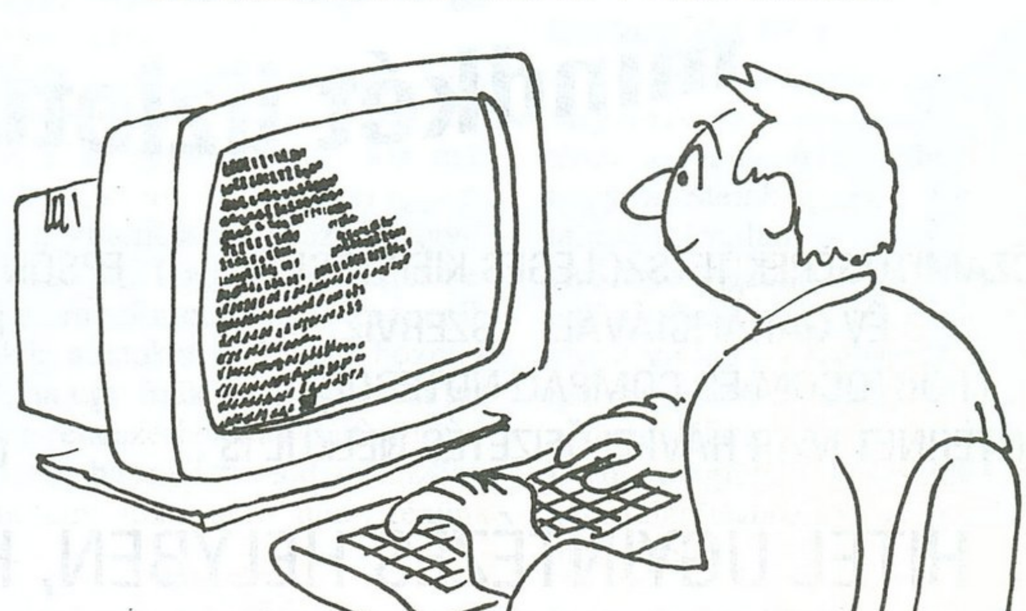
Megpróbáljuk?

Egyszer azután Windows alá kellett egy komolyabb programot írnom. Persze a Delphit választottam fejlesztőeszköznek, végül is ez a Pascal utódja, az én szempontomból ez is csak Pascal. Azt tudomásul vettem (pontosabban nem is gondoltam az elméleti háttérre), hogy programozási értelemben a képernyőre húzott vizuális komponensek is objektumok. Könnyű volt ezt nem észrevenni, hiszen a Delphi elrejtette a részleteket. Én konstruálom az objektumot, de ráhúzással, tehát ha van is mögötte programkód (de még mennyi van!), az nem látszik.

Amikor például egy szöveg színét menet közben megváltoztatom, a szintaxis olyan, mintha egy rekordváltozó egyik mezőjének adnék értéket (Form1.color := alGrey;), így nem tűnik fel, hogy üzenetet küldök az objektumnak. Meghívom egy módszerét (method), és ezáltal az objektum tulajdonságai (property) megváltoznak.

Más esetekben azért már kell egy kis önbecsapás ahhoz, hogy ne lógjon ki az „objektumista" lóláb, például akkor, amikor azt írom, hogy OpenDialog1.execute. De ezt is lehet szintaktikai érdekességnek tekinteni, és nem foglalkozni vele. A program a felületen elég egyszerű volt, néhány adatbeviteli képernyőn egyetlen személy néhány tucat adata került a gépbe, viszont ezekkel az adatokkal rengeteget kellett számolni. A számoláshoz tartozott még egy

SZUBJEKTUMORIENTÁLT PROGRAMOZÓ



alaptáblázat, amelyet paraméterfájlból töltöttem be. Bizonyos adatait ennek a táblázatnak nem tartalmazta a paraméterfájl, ezeket betöltéskor számoltam ki, illetve az egyén adatainak függvényében újraszámolódtak. Induláskor világos volt, hogy van két rekordom és egy csomó függvényem, amelyek a rekordok mezőivel dolgoznak. Technikai okokból egyébként ezeket a rekordokat különböző unitokban helyeztem el.

Ahogy a fejlesztés haladt, szembe kellett néznem egy-két problémával. A kisebb (bár akkor ez tűnt a nagyobbnak), hogy nincs rendes dátumbevitel a Delphiben. Simay Endre István kollégánk kérésemre összedobott egyet, azt forráskóddal kaptam meg, azóta is nagyon meg vagyok elégedve az eredménnyel. A Delphi szóhasználatában ez egy vizuális komponens, tehát installálni kellett, és azután ugyanúgy ráhúzogatni a képernyőre, mint a gyáriakat, ezért nem is foglalkoztam a belsejével. De a sors úgy hozta, hogy ez a négyjegyű évszámokat fogadó komponens kevés volt, kellett egy, amely kétjegyűeket, időtartamokat fogad. Nekiestem az eredeti komponensnek, és faragtam belőle egy másikat, amely a kétjegyűeket kezeli. Nem volt ebben öröklődés, de a forráskód átírása során mégiscsak szembesültem azzal, hogy vajon miként is működik az eredeti (meg az új, az általam készített), és kezdtem belátni, hogy ezek tényleg objektumok.

Lustaság, hibrid, tanulság

A nagyobbik probléma a lustaságom volt. Az egyedi adatrekord mezőivel kellett műveleteket végeznem. Egyszerűen csak „adatrekordnak" neveztem, de számtalanszor le kellett írnom, hogy melyik rekordról is van szó. Egy részük megspórolható volt a „with" direktívával, de azért ez is csak írás, másrészt időnként zavaró. Az objektum esetében a fordító tudja, hogy minek az adatmezőivel dolgozom. Lassanként, ahogy alakultak a rutinok, rájöttem, hogy legyen csak az adatrekord egy objektum, és a különböző mezőivel végzett műveleteket már mint az objektum eljárásait definiáltam. Egyszerre csak valamivel rövidebb és tisztább lett a kód, egyszerűbbé váltak a módosítások. (De azért tegyük hozzá, hogy ugyanezt a hatást majdnem ilyen könnyen elérhettem volna hagyományos módszerekkel is.)

Szórakozásból az alaptáblázatot „juszt sem" definiáltam át objektummá, így össze tudtam hasonlítani, hogy mit jelent írásban és kódolásban a különbség. Egyértelmű volt, hogy objektumos megközelítésben kevesebbet kellett írnom. Ráadásul az objektum (örökölvén a Delphi objektumok alaptulajdonságait) külön kiírható volt lemezre, visszaolvasható volt. Kaptam egy csomó kényelmi eszközt a fejlesztés közbeni objektumböngészéssel is, ezek jelentősen gyorsították a munkát. A végeredmény persze valami hibrid lett. De most már legalább eleve tudom, hogy a következő ilyen nagyobb munkánál objektumot fogok készíteni mindenből, ami arra érdemes...

Egy még személyesebb megjegyzéssel zárom. Ennek a cikknek a megírása közben gondolkodtam el mindazon, amit fejlesztéskor rutinból vagy ösztönből csináltam. Most kezdtem igazán megérteni, hogy mit is jelent az objektumorientáltság, és hogy a programban, amit írtam, hol is használtam ennek különböző elemeit. Nem lettem fanatikus híve, de el kell ismernem: nem is olyan rossz dolog.

Horlai János

Áldás vagy átok?

OO barbatrükkök

A programozásban igen divatos az OOP és mindaz, ami ezzel érkezett: az OCX (ActivX) kontroll és a COM/DCOM technológia. De érdemes mélyebben is elgondolkodni rajta: következményeiben nem hordoz-e veszélyeket az új irányzat.

A technológia objektumorientált megújulása sok könnyebbséget hoz a mindennapi munkában, hiszen ezek az építőkövek önállóan is jelen vannak. Sőt, mi több: segítségükkel — mivel nem kell a forráskódot kiadni — olyan ismeretek is megvásárolhatóvá válnak, amelyekhez korábban csak beépítve, egy-egy nagyobb programban juthattunk hozzá. Most pedig úgy építhetjük be saját termékünkbe a megvásárolt aktív kontrollt vagy COM modult, hogy esetleg azt sem tudjuk róla, mit is csinál valójában.

A felszín ismert, de vajon mit rejt a mély? Mi minden bújhat meg az ismeretlenség homályában? Lehet, hogy minden tisztességesen zajlik, de számolnunk kell azzal is, hogy — akárcsak a számítástechnika egyéb területein — nem mindenki tiszteli az etika szabályait. Márpedig ezen a területen sokkal több lehetőség nyílik rá, hogy kellemetlen meglepetésben részesítsék a mit sem sejtő felhasználókat.

Négy akna

A vírustechnológia is használ zseniális programozási trükköket, ezeknek a kiagyalása azonban eleve romboló szándékú, tehát már a kiindulópont is etikátlan. Az objektumorientált alaptechnológia megteremtése viszont kétségtelenül jó szándékú. Ettől azonban még felhasználható sötét célokra. Bizonyos értelemben nagyobb is a veszély, mint a „kisstílű" vírusok esetében. Ez az új technológia ugyanis nagyon hatékony.

Nem sokkal azután, hogy megjelent és használhatóvá vált a COM technológia és az OCX, már kezdtek felbukkanni az ezekre épülő „piszkos trükkük" (dirty tricks). Az eljárásnak négy olyan sajátosságát látom, amely kihasználható a felhasználó ellen:

1. Nem tudható, hogy mi van a modul belsejében. A felhasználó készen veszi a modult, és csak annyit tud róla, amennyit gyártója (és forgalmazója) szükségesnek látott közölni. Tehát tudatosan beépíthetők a modulba például „biztonsági lyukak" vagy speciális átjárók más programelemekhez.

2. Más alkalmazás is megteheti, hogy megszólítja ezt a modult, ha a létezéséről tudomást szerez, ha látja. Másképp kifejezve: valamilyen alkalmazással — akár egy demóprogrammal is — bejuttathatunk a rendszerbe egy olyan modult, amely más, korábban már meglévő alkalmazással elkezd beszélgetni. Önmagában ez nem volna baj, hiszen ezt a technológiát is fel lehet jó célokra használni. A veszélyforrás az, hogy ez a lehetőség alattomos célokra is alkalmazható.

Hasonlatként utalok a többkomponensű harci gázokra a katonai technológiában. Önmagukban ezek komponensei teljesen ártalmatlanok, sőt az sincs kizárva, hogy mindegyiket használjuk a mindennapi életben. Ha viszont keverednek, előáll a végeredmény, a harci gáz.

3. Az egyes elemek egymástól igen távol, és akár mások platformján, hálózaton át is futtathatók. Kinek-kinek a fantáziájára van bízva, hogy ez mi mindent jelenthet.

4. Végül pedig könnyű belátni, hogy a bejuttatott modul bármit meg tud csinálni, akár egymagában, akár más modulokkal, szoftverelemekkel együttműködve, különösen, ha azt is figyelembe vesszük, hogy a hálózaton keresztül elérhető modulokkal kooperálhat. Akár adatokat is ki tud hozni a gépből, ha úgy írták meg. Jelentést tud küldeni a rendszerben használt szoftverekről, azok használati módjáról. Mintha csak egy ipari kém ülne gépünk mélyén.

A nagy testvér figyel

Hogy mindez rémálom lenne? A közelmúltban két hír is megjelent a videoiparban:

a) A nagy médiacégek bejelentették azt a kötelezően beépítendő áramkört, amely megakadályozza, hogy a felhasználó részleteket vagy egész műsort vegyen fel saját videomagnójára, illetve hogy gépében DVD lemezről és sugárzott zenei vagy videoanyagról származó anyagot tároljon.

b) A számítógépes tévévevőkbe és videókba be kell építeni az Egyesült Államokban a cenzúra bizonyos formáját megvalósító eszközt, az úgynevezett V-chipet.

Lassan tehát kezd egyre inkább aktuálissá válni az amerikai hackerek által már évek óta hangoztatott felhívás, hogy „konzerváld a jelenlegi technológia eszközeit a jövő számára, mert lehet, hogy az új technológiák által egyre szorosabban ellenőrzött világban ezekre a régi eszközökre támaszkodva tudod majd megőrizni szabadságodat és függetlenségedet…".

A fekete doboz mint rejtekhely

Visszatérve a hónap témájához, a mostanság egyre jobban terjedő OCX-COM programozási és programfuttatási technológiához, nézzük meg, hogyan használják ki egyesek a fenti lehetőségeket.

A COM és az OCX elemeket gyártó cégek többsége nem él vissza azzal a helyzettel, hogy nem tudjuk, mi van a „fekete dobozban", ezért általában nyugodtan használhatjuk ezeket a sokat ígérő programelemeket. Az ő érdemük, hogy kiváló programrészletek kaphatók, gyakorlatilag filléres áron. De természetesen megjelentek a modulgyártó hiénacégek is. (Bár minél tovább nézegetem a Spectrum tévén az afrikai természetfilmeket, annál inkább sértőnek érzem ezt a hasonlatot — a hiénákra nézve.) És a példa ragadós.

Barbatrükk № 1

Az egyik barbatrükkös megoldásnak magam is szenvedő alanya voltam. Történetesen magamnak kellett rájönnöm, hogy a Steinberg cég egy látszólag teljesen ártatlan — és avatatlan szem elől rejtve maradó — OCX kontrollmodulba rejtette el digitális stúdiószoftverének hardlock-kezelését. Nem a készülék megjavítása, hanem a hiba okának a megtalálása volt itt a nagy dolog. Utána megfelelő fejlesztőkészlettel a hatástalanításhoz már elég volt néhány másodperc.

Barbatrükk № 2

Akárcsak az első vírusok megjelenésekor, az OO-trükkökben is a meglepetés okozza a kár nagyobbik részét.

Egyes cégek erre építve kezdenek most alkalmazni egy újabb aknát: OCX vagy COM objektumokban rejtik el a maguk eufemizmusával „authorizáló modulnak" nevezett programrészletet, amely bizonyos idő elteltével (vagy valamilyen más feltételre) egyszerűen kikapcsol. A demóprogramoktól az ilyesmit senki nem veheti rossz néven. A leselkedő nagyobbik veszély abban rejlik, hogy bizonyos szoftvergyártók — köztük a Microsoft is — már komolyan fontolgatják annak lehetőségét, hogy kereskedelmi programjaikba is beépítenek ilyen modult. Mi ebben a ráció? — kérdezhetik.

Ha a feladat időhöz kötött, akkor a program a megadott idő vagy indításszám után nem működik. Ilyenkor a program forgalmazójának arra is van módja, hogy bejelentse: az én termékem egyáltalán nem eladó. Csak bérelni lehet, korlátozott időre, adott indításszámra. És ha a program valóban jó, akkor megteheti, hogy jóval többet is kér érte, mint amennyit egy normálisan működő piac hajlandó lenne megfizetni.

A felhasználó ezáltal reménytelenül kiszolgáltatott helyzetbe kerül. Kulcsfontosságú programoknál megteremtődik a számítógépes szabadrablás lehetősége, kisebb jelentőségű szoftverek esetében pedig az árfelhajtás könnyen működtethető eszközéhez jutnak hozzá a gyártók. A gyanútlan felhasználó ma még könnyen besétál a kelepcébe, nem látja át, hogy jobban járna, ha ilyen feltétellel meg sem venné a programot. Van már hazai programfejlesztő cég is, amely így igyekszik maximalizálni saját profitját. (A pereskedés elkerülésére nem írjuk ki a nevét.)

Barbatrükk № 3

A Microsoft hasonló jellegű kísérletének az egész magyar piac szenvedő alanya volt. A cég emberei ugyanis olyan modult építettek be népszerű szövegszerkesztőjük otthoni használatra szánt változatába, amely ellenőrzi a gép konfigurációját, és annak megváltoztatása esetén arra kényszeríti a felhasználót, hogy az ügyfélszolgálattól új aktivizációs kódot kérjen, különben ugyanis adott számú — jelen esetben húsz — futás után a program leáll.

Ráadásul a „magyar kísérletben" alkalmazott módszert a legutóbbi szoftverjogi konferencián (amelyet stílszerűen az ELTE jogi karán tartottak) kifejezetten jó megoldásként emlegette a cég hazai jogásza, mondván, hogy az a felhasználók elégedettségével találkozott. Mi történik például, ha az ember kivehető merevlemezzel dolgozik? Minden merevlemezcsere után szaladjon aktivizációs kódért? Már maga a követett módszer alapfilozófiája is hamis, legalábbis itt, Nyugat- és Kelet-Európában. Itt az emberek folyamatosan változtatják a megvásárolt konfigurációt, szemben a dobozvásárló — és technikailag igencsak mulya — amerikaiakkal. Mint általában a védelmek esetében, itt is a felkészületlen jogos felhasználókkal tolnak ki leginkább. Ha ez a megoldás elterjedne, éppen a szoftverkalózok forgalmát növelnék meg jelentősen, hiszen legközelebb már az is hozzájuk fordulna, akinek ilyesmi különben esze ágában sem lenne.

Lehetőségből kényszer

Összefügg ezzel a kérdéssel az új technológiának egy másik, jóra is, rosszra is fordítható tulajdonsága: a hálózatos használat lehetősége. Erről is elmondható, hogy áldás is, átok is egyszerre. Áldás, mikor nyugodtan elviszem egy nagyobb gépre azt a modult, amelynek kiszolgálásához nem elég a saját gépem teljesítménye. A Linux, Solaris, valamint a Mac operációs rendszer felhasználói felülete a COM technológiához rövidesen bárkinek rendelkezésére áll az Interneten, méghozzá ingyenesen.

Ez a dolog jó oldala. De mi történik, ha egyes cégek felismerik a kínálkozó lehetőséget a felhasználók kiszolgáltatottságának fokozására, saját profitjuk-feltornászására?

Megtehetik ugyanis, hogy a kulcsfontosságú modulokat nem adják el, hanem folyamatos fizetés ellenében az Interneten letölthetővé teszik. Nekik csak az kell, hogy a felhasználó fizesse az egyre emelkedő bérleti díjat. Efféle megoldásokkal könnyen monopolszerephez juthat egy cég, különösen akkor, ha képes is rá, hogy mindezt hatékonyan megvalósítsa. (Gondoljunk csak a távközlési monopóliumokra. A tarifák ott és akkor csökkentek, ahol és amikor a monopolhelyzet megszűnt. Nálunk erre még néhány évet várni kell.)

A bérleti lehetőség egyébként meleg fogadtatásra talált néhány számítógépgyártónál. Az amerikai IBM sajtóközleményében olvashattuk, hogy már dolgoznak egy alaplapra szerelt integrált eszköz kifejlesztésén, amellyel a szoftverbérlet intézményesen megvalósítható. Tárolja a hálózatból vett jogosultsági adatokat, így nem kell folyamatosan a hálón lenni. A bérlet meghosszabbítását is automatikusan intézi, elég, ha a gépnek hálózati kapcsolata van egy hitelkártya-tranzakciós központtal.

Kémkedő modulok

A COM technológia hatékony eszköze lehet az ipari kémkedésnek is. Hiszen segítségével egy szoftverelem elterjeszthető például egy demóprogrammal. És egy ilyen regisztrált alkalmazást könnyen meg is lehet szólítani. (Telepítéskor mi akadályozná meg, hogy önmagát beírja a registrybe?) Annak sincs akadálya, hogy bizonyos (látszólag eltérő forrásból származó) komponensek majd felismerjék egymást, és együtt egészen mást is megvalósíthatnak, mint ami a hivatalosan bevallott funkciójuk. Nem vagyunk messze attól, hogy azok a korlátok is ledőljenek, amelyek a vírusok útját eddig még többé-kevésbé eltorlaszolták.

Az OCX és a COM nem idegen elem. A teljes rendszer visszafejtése nélkül nagyon nehezen deríthető ki, hogy nem hordoz-e más funkciókat is, mint ami a deklarált cél. Meglehet, hogy a gépben meglévő szoftverek felhasználásának lehetősége forradalmi újdonságokat hoz a programozásba, az pedig biztosra vehető, hogy a weblapokat hihetetlen mértékben feldobja, de hogy veszélyei szinte beláthatatlanok, az már ma is sejthető.

Kölcsönvett intelligencia

Erre is érvényes az a definíció, amelyet a víruskorszak hajnalán Buruzs Tamás közgazdász mint antivírus-szakember így fogalmazott meg: „A vírus intelligencia értelem nélkül. Intelligenciáját a programozójától kapta, és annyira erkölcsös vagy erkölcstelen, amennyire a programozója." Ismét új korszak köszönthet be a számítástechnikába: kialakulóban van az emberi értelemmel nyilvántarthatatlan sok milliárd apró modul világméretű birodalma.

Akárcsak a vírusok elterjedése nyomán, talán itt is nemsokára külön keresőprogramoknak kell felderíteniük az ártó szándékú hívatlan látogatókat. Csak kicsit nehezebb lesz a dolguk, mert a modulok azonosító számai emberileg már kezelhetetlenek. Végérvényesen elmúlt a hangzatos vírusnevek ebből a perspektívából kifejezetten szépnek látszó világa.

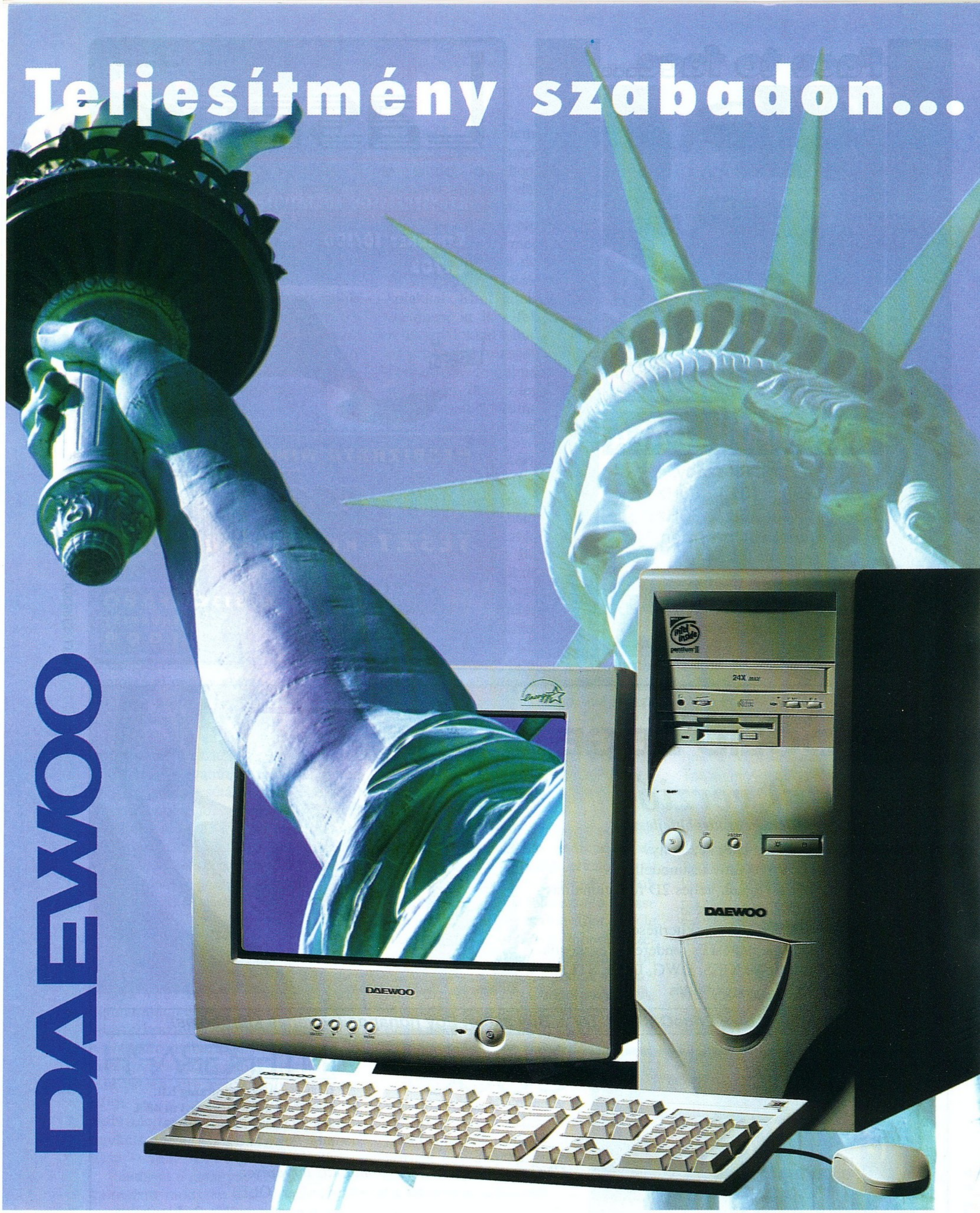
Jött valami új, valami más, aminek még csak a körvonalai bontakoztak ki. Annyi azonban már látszik, hogy a rosszat is jobban meg lehet vele valósítani, mint a korábbi megoldásokkal. Az új technológia áldás és átok egyszerre. Vajon melyik hatása lesz az erősebb?

Kis János


Face to face...
ADI
IFABO AKCIÓ
ADI 5P
87 000,- +ÁFA
17" MPRII szabv.
1280 x 1024 60Hz
1024 x 768 85Hz
Digitális/On-Screen
DATA KFT
1135 Budapest, Lehel út 58.
Telefon: 20/419-263; 20/362-382


LaserBit
COMMUNICATIONS
NAGYSEBESSÉGŰ ADATÁTVITEL LÉZERSUGÁRRAL
- Ethernet 10/100
- G.703
- Hang, kép, adat
- Token-Ring
MEGBÍZHATÓ MINDEN IDŐBEN!
HÓ - ESŐ - KÖD
TESZT KIÁLLÍTÁSUNKON
CROWN-TECH
A/313C
1118 Budapest, Pannonhalmi út 35. Tel.: 319 2995, 319 2996, 319 2997
Fax: 319 3326, Support Center: 319 3327
E-mail: crowntec@hungary.net www.crown-tech.hu
IFABO
BUDAPEST
1998


INTERGRAPH
SOLID EDGE™
4.0 verzió
150 új funkció és továbbfejlesztés
✔ Magyar nyelvű felhasználói kézikönyv
✔ 3D parametrikus, asszociatív testmodellezés
✔ Szabványos alkatrészrajzok, teljes 2D funkcionalitás
✔ Szerelési összeállítások tervezése
✔ Lemezalkatrészek tervezése és kiterítése
✔ Professzionális megjelenítési és renderelési eszközök
✔ Integrált CAD fordítók (IGES, DWG, DGN, STL)
✔ Microsoft Office 97-kompatibilitás
Solid Edge 4.0 és EdgeCAM 3.1 Try & Buy akció: Egy hónapos ingyenes kipróbálás!
edge CAM 3.1
INTERGRAPH
Intergraph Magyarország Kft.
1126 Budapest, Istenhegyi út 40/A
Tel.: 214-2007 Fax: 214-9588
http: //www.intergraph.hu
CNC megmunkálások automatikus tervezése:
✔ Közvetlen kapcsolat a Solid Edge-hez (OLE)
✔ 2,5–5 tengelyes marás, 2–4 tengelyes esztergálás, 2–4 tengelyes szikraforgácsolás, testreszabható poszt-processzorok
✔ EdgeCAM for Mechanical Desktop: integrált NC programozási felület az AutoCAD Mechanical Desktop 2.0-hoz
✔ Akciós megoldáscsomagok: Production, Professional, Mold


Teljesítmény szabadon...
DAEWOO
DAEWOO
DAEWOO
24X max

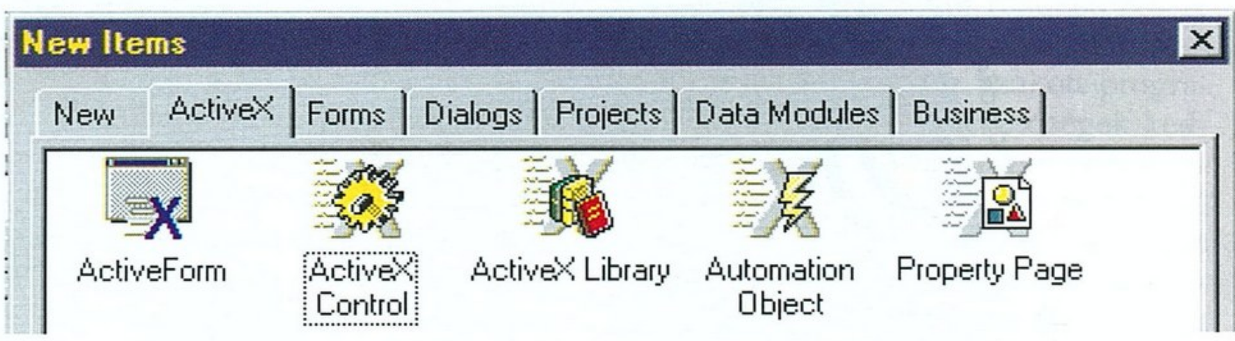
Presto, egy gyors lépés a jövő felé

Delphi 3.0 C++-ban

A vizuális fejlesztőeszközök piacán annak idején jelentős sikert aratott, sőt áttörést jelentett a 16 bites Borland Delphi 1.0. Ezt hamarosan követte a Delphi első 32 bites változata 2.0 verziójelöléssel. Ennek a verziónak jelent meg a C++ nyelvet támogató változata Borland C++ Builder néven, verziójelölés és fantázianév nélkül. Számos újdonsága mellett a Delphi 3.0 verzióját követő C++ verzióban ez is megváltozott: 3.0 jelöléssel és Presto néven jelent meg.

A Presto minimális rendszerigénye egy 486DX/100-as processzor, mintegy 170 MB merevlemez-terület és 24 MB RAM, valamint 32 bites Windows operációs rendszer. A tapasztalat szerint azonban a fordításkor létrehozott köztes állományok helyigénye miatt merevlemezből inkább 200 MB körüli üres terület javasolható, és bár elindul 16 MB RAM mellett is, a probléma nélküli futáshoz és a kivárható fordításhoz inkább 32 vagy még több szükséges.

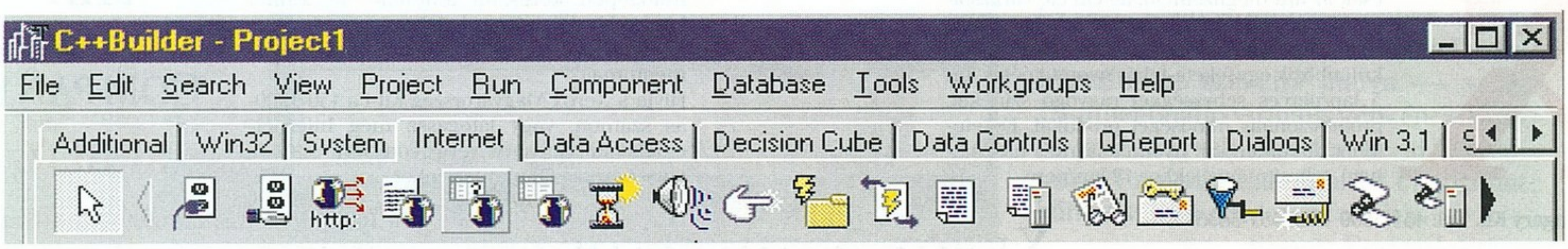
Az új C++ Builder, a Presto megalkotásával a Borland deklaráltan a Delphi 3.0-ban megvalósított lehetőségeket kívánta elérhetővé tenni a C++-ban fejlesztők számára. Természetesen megmaradtak mindazok a lehetőségek, amelyek a korábbi verzióban is rendelkezésére álltak a programozóknak. Így továbbra is lehetőség van személyre szabni a fejlesztőkörnyezetet, mind a szövegszerkesztő tulajdonságainak, mind a funkciók gyors elérését lehetővé tevő eszközsornak a tekintetében. Ezentúl azonban a gyors kódkiegészítés lehetősége adva van a C-ben fejlesztőknek is. Újdonság az is, hogy jelentősen bővültek a hibakeresés lehetőségei. Az új 32 bites debuggerrel (a 16 bites, DOS-os környezetben megszokotthoz hasonlóan) akár a CPU regisztereinek szintjéig megnézhetők a program működésének hatásai. Mód van a DLL állományokban is a hibakeresésre, programjainkban pedig a feltételes töréspontok definiálására.



A korábbi C++ Builder verzióhoz hasonlóan megmaradt annak a lehetősége, hogy a saját, korábban Delphiben fejlesztett 32 bites komponenseinket, ha forráskódjuk rendelkezésre áll, beépíthessük a programjaink számára elérhető komponensek közé. A forráskód alapján, a konverzió során a Presto saját Delphi-fordítóval dolgozik, így a Delphivel nem rendelkezők is felhasználhatják a szép számban, gyakran forráskódként is hozzáférhető komponenseket. A Pascal források beépítésekor a 32 bites Delphi 3 komponenseivel nem vallottam kudarcot, azok probléma nélkül átalakultak a Builder komponenseivé. Eközben a .DCU fájlok mellett megjelentek a használatukhoz szükséges tárgykódú fájlok (.OBJ) és fejlécfájlok (.HPP) is. Azonban a Presto által fordított .DCU unit fájlok eltérnek a Delphi 3.0 fejlesztőkörnyezetben generáltaktól, amit a különböző fejlesztőkörnyezetet párhuzamosan használóknak figyelembe kell venniük.

A Prestóban a korábbi Builder-verziótól eltérően, az abban generált egységes nagy komponenskönyvtár helyett a Borland fejlesztői a C-programozóknak is a Delphi 3-ból ismert „kiscsomag" technikát adták. Ezzel a technikával különböző komponensgyűjteményeinket külön-külön csomagokba installálhatjuk. Később pedig megadhatjuk ezek beépítésének mértékét a fordításkor, így a futtatható EXE-fájl mérete néhányszor 10 KB-ra is leszorítható, a bináris kódú csomagok pedig külső egységként kezelhetők. Szintén a komponenshasználatot érinti, hogy külső komponensforrásként nemcsak Delphiben vagy Prestóban fejlesztett komponenseket használhatunk fel, hanem Windows rendszerbe installált, bármely más programnyelven írt ActiveX-vezérlőket is.

De a Presto a már meglevő ActiveX-komponensek felhasználása mellett azok készítését is támogatja. Erre az új elemek készítéséhez segítséget adó varázsló szolgál, amelyből ennek során először az ActiveX elemekkel kapcsolatos lapot szükséges kiválasztanunk. Új ActiveX-komponens készítését választva meglévő, korábban akár Delphiből átemelt komponenseket is felhasználhatunk az új ActiveX-komponens alapjaként a Presto saját „gyári" komponensein kívül. Ezekből a fejlesztőkörnyezet automatikusan létrehozza a szükséges forrásfájlokat, és ha korábban nem készítettünk az új komponenst befogadó ActiveX-könyvtárat, az ennek létrehozásához szükséges fájlokat, illetve a későbbi felhasználáskor megjelenő névjegyablak prototípusát is.





Xerox DocuPrint C55
És a monitoron látható színek papíron is *elérhetővé* válnak.

XEROX ®, The Document Company és a stilizált „X"® a XEROX CORPORATION bejegyzett márkanevei. Az „After Dark Images" a Berkeley Systems Inc. bejegyzett márkaneve.

A Xerox DocuPrint C55 nyomtatónak nemcsak az ára megfizethető, de Ön elé varázsolja mindazt a színgazdagságot, amelyre szüksége lehet! A DocuPrint C55 ára nem sokban különbözik egy fekete-fehér nyomtatóétól, ám 3 lap/perces sebességgel ragyogó színeket, professzionális minőséget produkál. Fekete-fehér dokumentum nyomtatására is használható, teljesítménye ekkor 12 lap/perc.



A nyomtatót a hálózatvezérlő szoftverrel számítógépén keresztül állíthatja be. Ennél könnyebb dolga nem is lehet.
Szeretné színesben látni fekete-fehér dokumentumait?
Hívja a Xerox Magyarország Kft-t a 436-1900-as számon, vagy látogassa meg Internet-oldalainkat a www.xerox.com/new címen, vagy keresse föl partnereinket.

THI
DOCUM
COMP
XER

CHS Hungary Kft. Tel: 451-3500 Fax: 451-3535

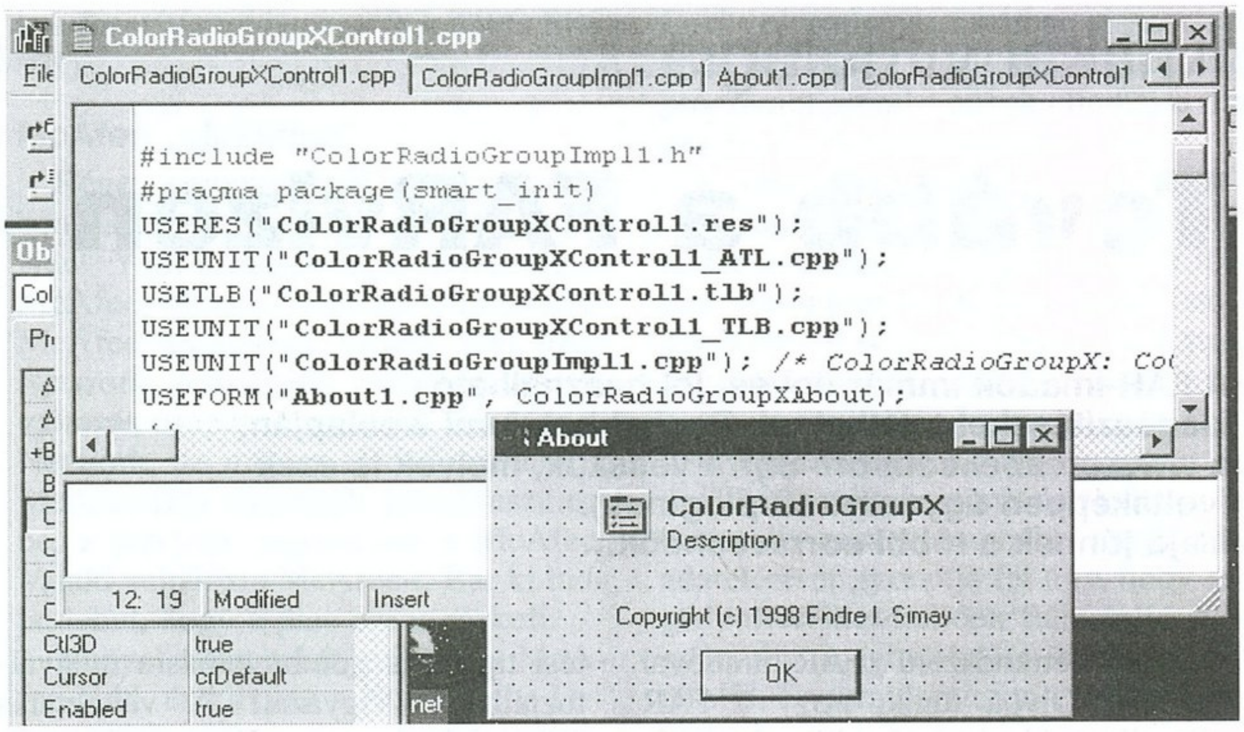
HRP Hungary Kft. Tel: 252-6300 Fax

XEROX NYOMTATÓK

Ezek segítségével később lehetőség van a komponenskönyvtár .OCX fájlba fordítására. Ezt a RUN menü alatt megtalálható opció segítségével regisztráltathatjuk a Windowsban, és a frissen regisztrált komponenst a többi ActiveX-komponenssel azonos módon a palettára helyezve egy új alkalmazásban ki is próbálhatjuk. De lehetőség van akár teljes Formokat ActiveX-elemekké fordítani, amelyeket később akár HTML dokumentumokba is beépíthetünk. Az ActiveX-es fejlesztés során jelentős segítség, hogy a regisztrálást meg is szüntethetjük, ami a Registry épségének megőrzése szempontjából nem elhanyagolható.

A Builder új verziója ezzel is a csoportos fejlesztést segíti. Ezáltal a Prestóban fejlesztett elemekkel más fejlesztőkörnyezetben végzett munkák is kiegészíthetők. Ez a közös fejlesztés természetesen kétirányú folyamat. Ennek két lehetőségéről, a Delphi-komponensek átvételéről, illetve az ActiveX-komponensek integrálásáról már esett szó. Azonban a Pascal-fejlesztők „termékei" közül nemcsak komponenseket építhetünk be alkalmazásainkba. Komplett, más célt szolgáló egységeket is beemelhetünk, sőt akár teljes Formokat is. Ez esetben az eredményes fordítás előfeltétele, hogy a Pascal-állományban használt valamennyi (nem Borland eredetű, és így a Prestóban is meglevő) egységet (Unitot) hozzá kell rendelnünk a Builder-projekthez, majd konvertálnunk kell. Korlátozás azonban, hogy például a Delphi-Form komponensei közül nem törölhetünk.

A csoportos fejlesztést segíti az is, hogy a korábbi verzióhoz hasonlóan különböző kiszerelésben forgalmazott program „magasabb" professzionális kategóriái komplex verziófigyelést és csoportmunka-támogatást tartalmaznak, ami most kibővült a projekt-csoportok kialakításának lehetőségével. Az egyedi projektek felvehetők a maguk alkalmazásaival és egyéb elemeivel, így ezekkel komplex és komplett hálózatos alkalmazásfejlesztések és Internet-alkalmazások készíthetők. Ezekhez rendelkezésünkre állnak mind a hálózati eléréseket segítő, mind a HTML formátumot támogató komponensek. A hálózatos elemek pedig, például egy webszerveralkalmazás építésekor azonos programozási felületen teszik lehetővé a különböző hálózati megoldások (ISAPI, CGI stb.) elérését, mivel a komponensek az alkalamazáson belül egy belső réteggel fedik át a különbségeket.



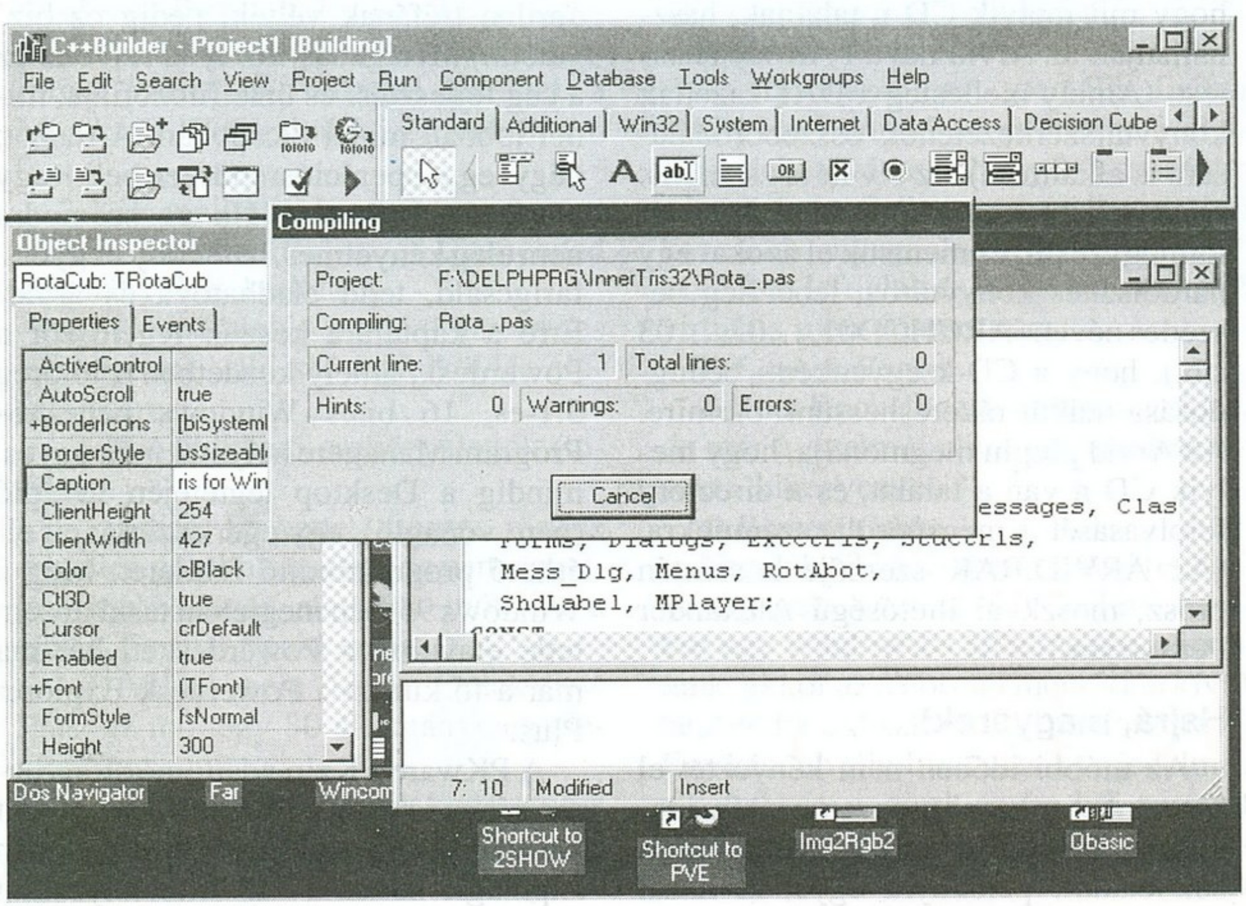
Mindazokat a lehetőségeket, amelyek a korábban piacra került Delphi 3-ban a pascalos világ rendelkezésére álltak, most — némileg ki is bővítve — megkapták a C++-hoz szokott programozók is. Az azonos lehetőségek kedveznek a programozói többnyelvűségnek is. És hogy a paletta teljes legyen, a Borland ígéri e programtermékből is az AS/400-asokhoz készült C++ Builder/400-at (a már rendelkezésre álló Java Builderhez és Delphihez hasonlóan).

Az újabb fejlesztőeszközöket látva, s tekintetbe véve, hogy időnként mennyire egy irányba tartanak, azon is elgondolkodhatunk, hogy mit fog hozni a jövő. A Pascalban és a C++-ban végzett fejlesztések közös nevezőre hozása érdekében a Borland nagyokat lép előre. A jövő talán már a forráskódok keverhetőségét is tartogatja — lehetőséget adva arra, hogy egy-egy eljárást a legjobban kézreálló nyelven írhassunk meg.

A programozói végtermékek körében a COM technika implementálásával a programnyelvekbe, programkörnyezetekbe is kinyúlhatunk, ha ott fogadni tudják az alkalmazott eszközt. Ezek között pedig egyaránt találunk programnyelvi fejlesztőeszközöket (C/C++, Pascal, Basic, Java stb.) és adatbáziskezelőket.

A másik út a C nyelv eredeti célját, a hordozható kódot megvalósító megoldás lehet a vizuális fejlesztőeszközök terén. De ezt — David Intersimone, a magyarországi termékbemutatót is tartó fejlesztési igazgató közlése szerint — a C++ Builder esetében a Borland nem tervezi. Megvalósulhat azonban a Java alkalmazásával, mivel — szintén David Intersimone szavaira utalva — az új JBuilder önálló virtuális gépként 100%-os Java-megvalósításként kerül fejlesztésre. Ez pedig már átjárást biztosíthat bármely más rendszer felé is.

Simay Endre István



Kritikai vitriológia

Tovább a FARvizen

A FAR-imádók immár ügyes, jól használható kiegészítéseket találhatnak Roshal belgiumi weblapján, a www.creabel.com-on. Sorra vesszük, melyek is ezek (voltaképpen úgynevezett pluginok). Majd jönnek a többi sorraveendők...

A menetrendszerű FAR-híradóval kezdem. Olybá tűnik, hogy a FAR világsikere kissé merész következtetés volt részemről, de menetelése azóta is töretlen. Mivel a szerző, a cseljabinszki Eugene Roshal teljesen nyilvánossá tette (sőt forráskódban közölte) a FAR Manager szoftverkapcsolódási felületét — pontosan abból a célból, hogy a hozzá hasonlóan ügyes kezű és ötletes programozók további hasznos bővítésekkel tehessék le a névjegykártyájukat a FAR mellé —, újabb gyöngyszemek bújnak elő, mint eső után a gomba.

A FAR-hoz máris készült CWI kódtábla (Addon). Ennek segítségével gombnyomásra konvertálhatunk Windows ANSI-vá vagy a 852-es kódtáblával kompatibilis állománnyá. Telepítése a pluginekéhez hasonlatos, csak ezúttal a ...\FAR\ADDONS\ könyvtárban csomagoljunk ki.

Egy plugin telepítése pofonegyszerű, a FAR\PLUGINS\ könyvtárban létrehozunk egy tetszés szerinti újabb directoryt, abba bemásoljuk magát a plugint (általában egy DLL-t, a hozzá tartozó helpfájlt és a *.REG Registry-bejegyzéseket), majd pedig újraindítjuk a FAR-t, mely meglátja és megszereti az új plugint, és F11-re máris ismerkedhetünk az új segédprogrammal.

• ***Registry Browser***

Az ötlet elementáris. Rengeteg Registry-röcögtetővel volt már dolgom, de Peter Norton, a Quarterdeck, a Nuts&Bolts és a Microsoft együttvéve sem volt képes hasonló frappáns, magától értetődően szemléletes megközelítésre, mint Michael Yutsis. (Nyomulnak a tehetséges orosz programozók!) A Registry innen nézve úgy kezelhető, mint egy terjedelmes virtuális könyvtárstruktúra, minden editálható, könnyedén kereshető, sőt megtalálható. Maguk a registry-kulcsok, azaz bejegyzések felfoghatók akár fájl-, akár directory-bejegyzésként, ahogy számunkra kényelmesebb. A bejegyzés megjelenítési típusa is gombnyomásra megváltoztatható. Vigyázat! A változtatás azonnal érvényre jut! Hozzáértő kezekben nagy kincs, de ha a konok nemtudás kellő önbizalommal párosul, jóvátehetetlen károk forrásává is válhat a könnyelmű kísérletezgetés.

• ***FCOMPLET.RAR plugin***

A 4DOS-ból (sőt azt megelőzően az Anarkey nevű parancshéjból) megkedvelt szolgáltatás, amelyet leginkább a Windows rámutogatós módszerétől ámulatba nem esett, a parancssori írásbeliséget megtartó emberek fognak értékelni. Ott ugyebár elegendő volt egy fájlnévből akár a kezdőbetű, és gombnyomásra az összes így kezdődő fájlnéven végigpörgött a rendszer, nem kellett például az elgépelések miatt aggódnunk. A 4DOS-ban a Tab és Shift-Tab gomb szolgált a névpörgetésre.

• ***ARVID — a zseniális***

Ugye elgondolkodtak már azon, hogy archivált CD-iket képtelenség lesz majd kezelni, ha nagyon felgyülemlenek. Ha már nem emlékeznek pontosan, hogy mit melyik CD-n találnak, használhatják az Arvid nevű FAR kiegészítést. (Amely mellesleg a DAT kazetták könyvtárszerkezetének összeboronálására is alkalmas.) Az elv itt is zseniális. Olvassuk be egyenként CD-ink tartalomjegyzékét, és mentsük el azokat egy harddiszkes könyvtárba, lehetőleg beszédes néven (ARCHCD01, ...02, ...03 stb.), hogy a CD-k egyenkénti bedugdosása nélkül rákereshessünk bármire. Az Arvid plugin megmondja, hogy melyik CD-n van a találat, és a directory felolvasását is megspórolja számunkra. (Az ARVID.RAR szerzője a szintén orosz, moszkvai illetőségű Alexander Petroszjan.)

Hajrá, magyarok!

Az utóbbi időben nem kényeztet el bennünket a hazai programozótársadalom. A hazai piac szűkös mérete miatt sok eladott példányra egyetlen hazai szerzőcsapat sem számíthat. Néha azonban felbukkannak „slágergyanús" alkalmazások. A gépjármű-költségelszámolás területén például eddig egyetlen olyanról sem volt tudomásom, amely véletlenszerűen megadott két település közötti távolságot kapásból kiszámolta volna, ha csak főutakat választok, vagy elkerülöm a fizetős autópályákat. Most végre született valami ilyesmi! (De ez most nem áprilisi tréfa, mint amit április 1-jén megjelent legutóbbi számunkban olvashattak a nem létező ConfigSYS Kft programjáról, a Titkárról.)

A hasonló témájú és Vándor nevű korábbi programra „rímeltetve" a Topo-Press Térképészeti Kft. elkészítette a Wandor-Boot windowsos programot, amely vektorosan tartalmazza Magyarország közúti hálózatát és összes települését. A program egyszerű kezelhetősége nagyszerű. Semmi sallang, semmi öncélú cicoma, csupán hazánk digitális térképe. A program jelentősen felgyorsul, ha nem íratunk ki minden részletet (például útszakaszok hosszát, kisebb települések nevét, nagyvárosok bel- és külterületeit, utak nevét, kompátkelőket stb.). Eddig a felhasználók csak a Microsoft által turisztikai célzattal kiadott angol nyelvű, meglehetősen szellős és adatokban szegény autóatlaszában próbálták megtalálni kis hazánk idevonatkozó adatait, de többnyire szomorúan csalódtak. Most a Wandor-Boot egyetlen földrajzszerető autós winchesteréről sem hiányozhat.

„Láb-mustra"

A Mijenix cég nem először szerepel cikkeimben. (Sokan ezt a cégnevet is áprilisi tréfának vélték, pedig ez biz' valódi.) Mi tagadás, közel áll szívemhez a cég fejlesztési és üzletfilozófiája: minél jobbat, minél olcsóbban. A recept: végy egy operációsrendszer-beli vagy egyéb szolgáltatást. Ha az számodra nem elég kényelmes, hatékony és gyors, farigcsáld, tedd eladhatóvá és áruld! Erre a kaptafára készült legelőször a Powerdesk, amely kezdetben a jó öreg 3.1-es, 16 bites Windows nehézkes Program Managere helyett kínált ügyes, mindig a Desktop legtetején lovagló (nem vonagló), egy egérkattintással elérhető programindító felületet. Még a Windows 95-ben megjelent taskbar sem tette elavulttá a Powerdesket, bár ma már a fő kunszt a Powerdesk Explorer Plus.

A PKware PKzip adattömörítő programjához is csak a PKware nem csinált kényelmesebb héjat. A Mijenix azonban Zipmagic néven a ZIP állományt mint

egyszerű foldert kezeli; abból akár programot is elindíthat az ember, a Zipmagic a folytonos frissentartást elfedi a felhasználó elől.

Erről a tőről fakadt a Freespace nevű program is, amely vígan übereli a Microsoft Drivespace-ét vagy a Stac cég hamvába holt Stackerét, és NT alatt is működik (30 napig futó kipróbálási változat feltalálási helye a www.mijenix.com.) Szóval ily módon ügyeskedik a Mijenix az 50 dollár alatti árkategóriában.

Ínyenceknek

Ezúttal a PowerDesk Utilities 98 fedőnév alatt megjelent PowerDesk v3.0-s változatát boncolgatnám egy picit. Korábban már ódákat zengtem róla, így most csak az ínyencségeket említem:

— Folder-szinkronizáció és -összevetés: ezen van ugyan még mit faragni, de kezdetnek nem rossz. A fájlok tartalmába egyelőre nem ás bele.

— Újabb tömörítők natív támogatása: RAR (ennyi lenne csak Roshal karrierje?!), ZOO, PAK, ARC.

— Automatikusan szeletel, ha nagyobb méretű archív állomány jönne létre, mint amekkora a floppy mérete.

— Archív állomány jelszavának kezelése.

— Szuper: az egyébként idétlenül semmitmondó névvel illetett szolgáltatás, a Dialog Helper vagy hosszabb nevén Open/Save Dialog Enhancement. Lényege, hogy megnöveli az Open/Save ablakok méretét, sőt megfejeli egy History Trackerrel. Ez utóbbi két újabb kis bigyónyi ikonkát eredményez az Open/Save soron az ablak jobb felső sarkában, ahol a korábban használt fájlneveket és könyvtárneveket tárolja, jelentősen túllépve például a Microsoft Word által kezelt maximum 9-et.

— E-mail-attachmentként kapott MIME, Base64, BinHex és Uuencoded állományok dekódolása.

— Uuenkódolás például internetes hírcsoportra való szövegfelküldés elősegítésére.

— Elmozdítható eszközpaletták (toolbar).

— Erase Diskette: floppy törlése egyetlen gombnyomásra, de nem egyetlen szemvillanás alatt (ez utóbbira a Quick Format opció szolgál).

— Akinél nem lenne QuickView Plus, az mintegy 80 állománymegjelenítőt kap.

— Grafikus állományok konverziója (például BMP-ből GIF, akár kötegelve, egy menetben többet is). 24-féle grafikus formátumot ismer (JPG, TIFF, EPS stb.).

McAfee „alakítása"

Régen szerepelt már seregszemlémben a Helix Software. Ezt a büszke kis New York-i magáncéget felvásárolta McAfee, akinek vállalata immár nem McAfee Associates, hanem NAI, azaz Network Associates Incorporated. A változás csupán nüansznyi, McAfee = Network, de míg az elsőről azonnal a McAfee-féle vírusölők jutnak eszünkbe, a NAI-ról legkevésbé a McAfee. Valaki behálózta McAfee-t. Bár ki tudja, lehet, hogy a piac ezt is mekefi.

Nagyot akart tehát gurítani a McAfee-NAI, és kihozta a Nuts&Bolts Deluxe változatát, ami tartalmát tekintve mindenképpen jó vételnek ígérkezik, hiszen másfél pénzen megkapja az ember a Norton Utilities-szerű NB 1.05-öt, a Hurrican 98-at, a McAfee VirusScan 3.1.4-et, a WebScant és a PGP-t (a Pretty Good Privacy nevű megfejthetetlen titkosítót). Ebből a programkavalkádból ezúttal csak a Hurrican 98-ról tennék néhány észrevételt.

Már az is nagy szó, hogy eddig teljesen működőképesnek bizonyult, pedig a Hurricane tizenhárom ponton veszi magának a bátorságot, hogy betüremkedjék a Microsoft és a Windows 95 fogaskerekei közé kenőanyagnak. Ezek rendre:

1) Boot Rocket

Mindnyájan tudjuk, hogy a Windows lassan és nehézkesen indul. Ezt az időt rövidíti le a Hurricane azzal, hogy a szükséges dolgokat begyűri egy bugyorba, és azt indításkor cache-elven azonnal betölti, amivel naponta (többször) fél percet is nyerhetünk.

2) Cache Compressor

A Hurricane egy speciális fizikai RAM-ban történő adattömörítési eljárással lehetővé teszi, hogy 2-3-szor több adat férjen a fizikai memóriába, ezáltal kevesebb anyagnak kelljen „átlógnia" a diszken lefoglalt és lassabban működő virtuális memóriaterületre.

3) Launch Rocket

A Boot Rockethez hasonló megoldás, de már a gyakran elindított alkalmazásokra kihegyezve.

4) Reverse Cache

Legalább 32 MB memóriával rendelkező gépeken van igazán értelme. Amikor egy windowsos alkalmazást elindítunk, akkor az a normál memórián kívül betöltődik a diszk-cache memóriába is. Egy később elindított másik terjedelmes alkalmazás a diszk-cache-ből viszont kitolhatja az elsőt. Ha ebből a másodikból kilépünk, a Hurricane a Reverse Cache elve alapján azonban visszarántja a helyhiány miatt kiebrudalt első programot, mintha annak elindítása óta nem is történt volna semmi.

5) Smart Read Ahead

Öntanuló algoritmus, amely elemzi alkalmazásainknak a harddiszkhez történő fordulását, és rutinos „gondolatolvasóként" viselkedik, pedig csak stréber diákként előre elolvassa a tananyagot.

6) Top Down Cache

CD-olvasók, network drive-ok, ZIP-drive-ok, azaz a lomhább meghajtók adatelérését gyorsítja fel úgy, hogy segédpuffereiket kellően felhízlalja és a gyorsabb meghajtón (a harddiszken) tárolja. És természetesen ez is stréberkedik.

7) RAM Expander

A Cache Compressorhoz hasonló módon, de a processzor „holt idejében" végez memóriazsugorítást.

8) Winpack

Az éppen nem aktív alkalmazások, taskok memóriabeli összepréselését szolgálja, hogy azok feleslegesen ne terpeszkedjenek, ha nem feltétlenül indokolt.

9) Mode Switch Accelerator

Azoknak jó, akik rendszeresen indítanak DOS-os alkalmazásokat is a Windows 95 alól. A processzor üzemmódváltásainak holtidejét csökkenti viszonylag kis mértékben.

10) Global DOS Memory Manager

A Windows előszeretettel piszkít a normál 640 KB és 1 MB közötti memóriaterületre. A Hurricane ebbe avatkozik be, és csak akkor ad szabad utat ebbe a régióba, ha valóban DOS-os programok kívánják azt használni. (Ne feledjük el, hogy a proci real módja annak nem a leggyorsabb működési állopota.)

11) Print Cache

Három sebességfokozata van. A nyomtatási munkálatokat egyre jobban a háttérbe kényszeríti, ami látszólag azt eredményezi, hogy gyorsabban folytathatjuk a munkát egy korábban kiadott printelési parancsot követően.

12) RAM Drive

Ügyes. Helyet csak akkor foglaló virtuális — de RAM-sebességű — diszk, ha benne ténylegesen csücsül adat. A leghasznosabb akkor, ha az egyébként diszkre ülő TEMP directory helyét ide kérjük bejegyeztetni.

13) XMS Accelerator

XMS-t is igénybe vevő DOS-os programoknál van jelentősége, a futási idő mintegy 10%-os lerövidülését eredményezheti.

Herczeg József

A debugger kivételes intelligenciájú, de...

A szoftverjog lángpallosa

Az Új Alaplap 1998. februári számában szerepelt a Perl programozási nyelv, amely, mint tudjuk, ingyenes. Teljesen ingyenesen, forráskóddal együtt hozzáférhető. Általában a Perl modulok, segédprogramok és hasonlók a jól koordinált CPAN archívumon keresztül szintén ingyenesek és szabadon hozzáférhetők. Noha a nyelv unixos eredetű, régóta léteznek DOS és Windows alá készített változatai is. Ezek közül a legelterjedtebb, és szinte szabvány az Activestate által írt változat. (Ez került fel a lap CD-mellékletére is.)

Az Activestate céget régebben Activeware, még azelőtt Hip communications néven ismerték. Általunk nem pontosan ismert, de szoros kapcsolatban van az O'Reilly kiadóval, amely elsősorban magas színvonalú unixos (és mostanában már linuxos, windowsos) szakkönyveiről ismert.

Február elején kiderült, hogy egy nagyobb projekt keretein belül (Visual Perl, brrrrr, még hallani is horror) készítenek egy Perl debuggert, nyomkövetőt. Az első bétaváltozat letöltésekor fel lehetett iratkozni egy levelezőlistára — meg is tettem —, ahol a hibákat, információkat csereberélik az emberek, és reagálnak az észrevételekre a szerzők.

A debugger kivételes intelligenciájú és jól használható volt. Tudni kell, hogy a nyelvben van egy beépített nyomkövető is, de az olyan mértékben „hajaz" a unixos eredetre, hogy csak a legelszántabbak használják, inkább jó programokat írnak, ami egyébként a Perlben nem is olyan nehéz.

Február végén megjelent a második béta, ez stabil volt, és még többet tudott, majd márciusban jött az értesítés, töltsem le a 3. változatot, mert ki akarják próbálni a licencrendszert, magyarul a végleges változathoz a védelmi, fizetési szisztémát.

Úgy tettem, ahogy mondták, de igencsak elcsodálkoztam. A megoldás egyrészt szellemes, mert Internet-kapcsolaton keresztül csinál valamit vagy a programmal, vagy a Registryvel, másrészt elborzasztó, mert miért kell nekem számomra átláthatatlan módon beengednem a gépembe valakit. Ráadásul ez a próbalicenc csak 3 napig érvényes, és azután ugyanezzel a technikával még kétszer 3 napra meghosszabbítható. A 3 nap abszurdnak tűnt, a legelvetemültebb próbaváltozatok sem szokták 14 napnál alább adni, de inkább a 30 nap a tipikus.

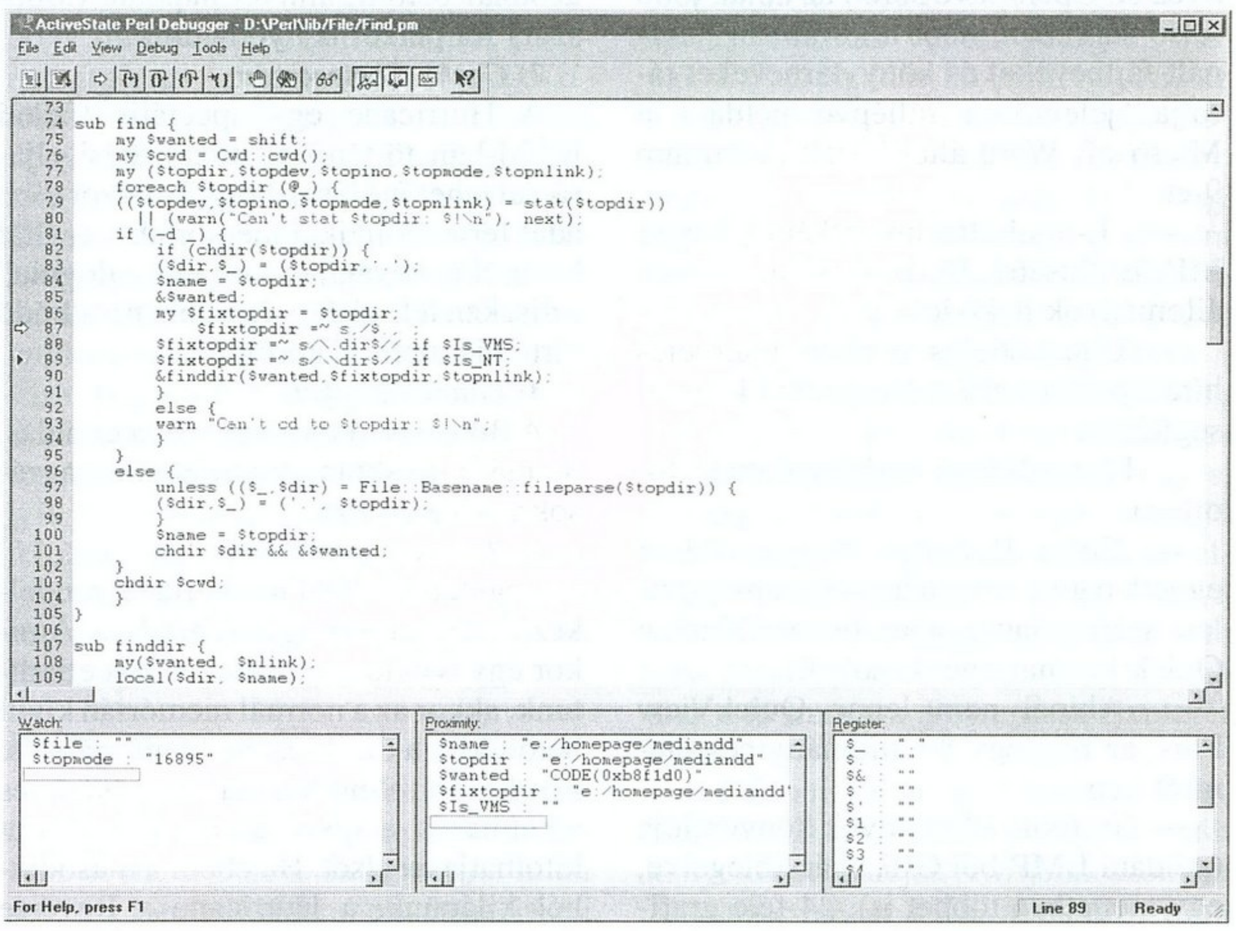
Mindezek után szemrebbenés nélkül visszatértem a 2-es bétához, amely április elejéig jó, és vártam a fejleményeket. Nem is maradtak el. Soha még ilyen érdekes és heves levelezővitát nem láttam, mint ami a licencrendszer körül kialakult. Kiderült, hogy az Activestate gépnévhez kötött, 49 dolláros, 3 napos próbaidejű megoldást akar, de tervez egy CD-n kibocsátott site/cég változatot is, limitált licencszámmal, azonban szintén a gép nevéhez kötve.

Az első hozzászólók egyike egy nagy szoftverháznál dolgozik, és azt fejtegette, hogy 80 fejlesztőjéből egyszerre 4-5 dolgozik perles munkán, de mindig másik 4-5. Ha mind a 80 emberre licencet kell venniük, az teljesen abszurd. Vagy mindig rakosgassák a gépeket, aszerint, hogy éppen kinek kell Perlben dolgoznia? További körülbelül tíz levél ugyanezt a problémát tárgyalta változó hevességgel.

A másik jellegzetes hozzászóló az az egyéni fejlesztő, aki azt mondja, hogy van egy gépe otthon, van egy laptopja, amelyet magánál hordoz, és amelyet egyébként 3-6 havonta jobbra cserél, de van egy harmadik gép is az állandó munkahelyén. Akkor neki most 3 licenc kell? Sokan azt is írták, teljes joggal, hogy egyszerre csak egy helyen vannak, ergo egyszerre csak egy ember használja a szoftvert. A hozzászólók azt taglalták, hogy szokatlan és érthetetlen a géphez, illetve gépnévhez kötött használati jog, mert általában fejlesztőhöz, emberhez kapcsolják a programok használati jogát.

Mások az elképzelés abszurditását illusztrálandó kifejtették, hogy laptopjukkal ügyfeleikhez járnak, ahol minden esetben más és más felhasználói névvel, illetve gépnévvel kellene bejelentkezniük.

A szoftvercég reakciójából kiderült, hogy ők úgy képzelik, mintha ez a nyomkövető egy operációs rendszer lenne, az is géphez kötött jogon működik (már ahol). Megértésüket mutatva



felvetették a gépnév helyett a felhasználói névhez kötöttséget, illetve egy 150 dolláros fejlesztői változat lehetőségét, sőt, ez utóbbiról még szavazást is kezdeményeztek. A válaszok nem is maradtak el... A legtöbben kifejtették, hogy mi a különbség az operációs rendszer és a nyomkövető között, a szavazásra pedig általában a „rossz kérdésre nincs jó válasz" szellemében reagáltak. Többen megemlítették, hogy egyetlen gépükön is többféle operációs rendszer van, ami manapság tényleg nem ritka — NT, Win95 meg egy béta Win98 nem is számít különlegességnek.

A vitában előkerült a múlt is. Hivatkoztak a Lotusra, amely kezdetben kulcslemezzel védte programját, gyakorlatilag lehetetlenné téve annak használatát. De hivatkoztak a Borland „no nonsense" licencszabályára, hogy kezeljük a szoftvert könyvként, amit egyidejűleg csak egy ember használhat. Tény, hogy a Borland egykori Turbo Pascalos sikerének egyik kulcsa az alacsony ár és a teljesen szabad (értsd: technikailag akadálytalan) másolhatóság volt.

Érdekes, hogy a közel 200 levél nagy része végig a jog talaján maradt, senki nem mondta, hogy ő ellopni, jogellenesen használni szeretné a nyomkövetőt, a jellemző hang az volt, hogy reális áron reális feltételekkel akarnak fizetni, és ha ez nem megy, akkor nem fogják használni a szoftvert.

Volt, aki arra is hivatkozott, hogy a Perl alapvető ingyenessége sem indokol ilyen szigorú és ostoba, a felhasználót eleve korlátozó politikát, és hogy a nyelv elterjedtsége, a perles társadalom segítőkészsége alapján nem így kellene profitot termelni, bármilyen jó is a szoftver, bármilyen sok munkát is fektettek az elkészítésébe.

Mellesleg többen megjegyezték, hogy a védelemmel kapcsolatos erőfeszítéseket inkább a program javítására használhatnák. Néhányan a klasszikus érvekre hivatkoztak, hogy a védelem szinte bármilyen fajtája a többségben levő becsületeseket gátolja, a kevés gazembert viszont nem tartja vissza. Persze volt egy-két igazi realista hozzászóló is, akik jelezték, hogy azoknak, akik netán lopni akarnak, nem lesz nehéz dolguk, mert az Interneten úgyis feltűnik majd a feltört program vagy a feltörést végző kód. Tapasztalataim szerint ez a feltételezés nem megalapozatlan, egy kis igyekezettel a piacon levő szoftverek 90%-ához megtalálható a feltörési segédlet.

Az Activestate-nek volt még egy erőtlen próbálkozása, hogy akkor név és társadalombiztosítási szám kombinációhoz kötnék a jogokat, mondván, a tb-számát (Amerikában) senki nem adja ki illetéktelennek. Erre lett csak felzúdulás, mivel a céget is érthető módon az ilyesmire illetéktelenek közé sorolták. Egy norvég pedig szerényen megjegyezte, hogy nem mindenki él az USA-ban, és az amerikai tb-szám nem világszabvány.

Itt tart most a vita, kíváncsi vagyok, vajon a józan ész győz-e, és egyszerű, újrainstallálásokat túlélő kulcsfájl- vagy sorozatszám-alapú védelem lesz-e a technikailag ötletes, de egyébként abszurd módszer. Én személyesen a 49 dolláros árat gondolkodás nélkül kifizetem, ha a program jól használható, de egyébként inkább maradok a sororientált, vacak, de ingyenes beépített nyomkövetőnél.

Azt hiszem, ez a levelezési vita sok más cégnek és a szoftverjog lángpallosú apostolainak egyaránt tanulságos lehet.

Horlai János

Fejlesztőeszközök

Objektumkezelés Linux alatt

DOS és Windows rendszeren nevelkedett programozók első nekifutásra talán idegenkednek a unixos fejlesztőeszközök spártai egyszerűségétől — itt bizony nincsenek (pontosabban még nincsenek...) grafikus felületek, varázslók és társaik. Akkor mégis mivel magyarázható, hogy rengetegen fejlesztenek Unix platformokra, különösen pedig Linuxra? A titok nyitja, hogy elképesztő mennyiségű fejlesztőeszköz áll mindenki rendelkezésére. Sok forráskód szabadon hozzáférhető, és tetszőlegesen módosítható, s maga az operációs rendszer is erre van kihegyezve. (Azt a tréfás mondást persze nem árt néha felidézni, hogy a Unixot programozók írták programozóknak, és a Unix ezt folyton érezteti is környezetével...)

Ha már elegünk van a C-ből, Assemblyből és a spagettikódból, tegyünk egy kis kitérőt az objektumorientált nyelvek világába! A Linux itt sem fog csalódást okozni. Aki kellő kitartással turkál a hálózaton (illetve a CD-n), az hamarosan a bőség zavarával küszködhet. A megszokott nyelvek mellett találunk különleges, más platformokon nem nagyon használt, vagy egyáltalán nem is ismert nyelvekhez is fordító és értelmező programot. Nézzük meg sorban, mit találhat az olvasó a mostani CD-melléklet Linux könyvtárában.

Fordítók

A megszokott „jó öreg" GNU C++-fordító, és a hozzá való osztálykönyvtárak (gcc-2.8.1.tar.gz, libstdc++-2.8.1.tar.gz, libg++-2.8.1.tar.gz). A csomagban természetesen benne van a 2.8.1-es C és Objective-C fordító is. Ez utóbbi a C nyelv egy érdekes és kevesek által ismert dialektusát „beszéli" — talán a GNUStep az egyik legjelentősebb projekt, amely ezt a nyelvet használja. Akinek tetszik a NeXT OpenStep felülete, és kíváncsi egyéb részletekre, nézze meg a http://www.gnustep.org/ címet!

Java

Korunk slágertémájából a Linux sem maradhat ki! Nemcsak a teljes fejlesztőkészlet (JDK 1.0.2-es és 1.1.5-ös verzió) került rá a CD-re, hanem két alternatív fordítóprogram is (guavac_1.0.tar.gz, kaffe-0.9.2.tgz). A Kaffe különlegessége, hogy a Java bájtkódot futásidőben fordítja át az adott platform természetes kódjára, jelentősen növelve ezzel a Java programok futási sebességét (természetesen nem csak Linux alá létezik).

Python

Sokak szerint a Perl trónját döntögeti, mások szerint csak egy újabb scriptnyelv. Ahogy a dokumentációjában olvasható: „a Python egy interpretált, interaktív, objektumorientált programozási nyelv. Gyakran hasonlítják a TCL-hez, a Javához, a Scheme-hez, a Perlhez. Moduláris, lehetővé teszi objektumosztályok, exceptionök, dinamikus adattípusok használatát, valamint csatlakozási felületet biztosít sok rendszerhíváshoz és ablakozó rendszerhez vagy grafikus felülethez." Erősségéről sokat elárul, hogy a Redhat disztribúció jó pár rendszerkarbantartó scriptje erre épül, valamint CGI programnak is sok helyen használják. A CD-re csak egy alapcsomag került fel, benne az interpreterrel és néhány modullal. Mindent összeszedni a Perlhez hasonlóan elég nehéz lett volna. Aki többet szeretne megtudni róla, és van Internet-elérése, az nézze meg a http://www.python.org/ címet!

XWPE fejlesztőkörnyezet

Az Xwindow Programming Environment révén rögtön cáfolom is, amit a cikkem elején írtam, ugyanis sikerült beszerezni egy integrált fejlesztőkörnyezetet. Aki használt már DOS alatt valamilyen Borland terméket, annak rögtön ismerős lesz, s ez nem véletlen: az alkotók igyekeztek az ott megszokott külalakot és viselkedést teljes mértékben másolni, hogy ez mennyire sikerült, döntse el mindenki maga! Az XWPE igen nagy előnye, hogy többféle nyelvhez (fordítóhoz) hozzáidomítható, és ismer háromféle debuggert is. Működik mind karakteres terminálon, mind X alatt.

Modula-3

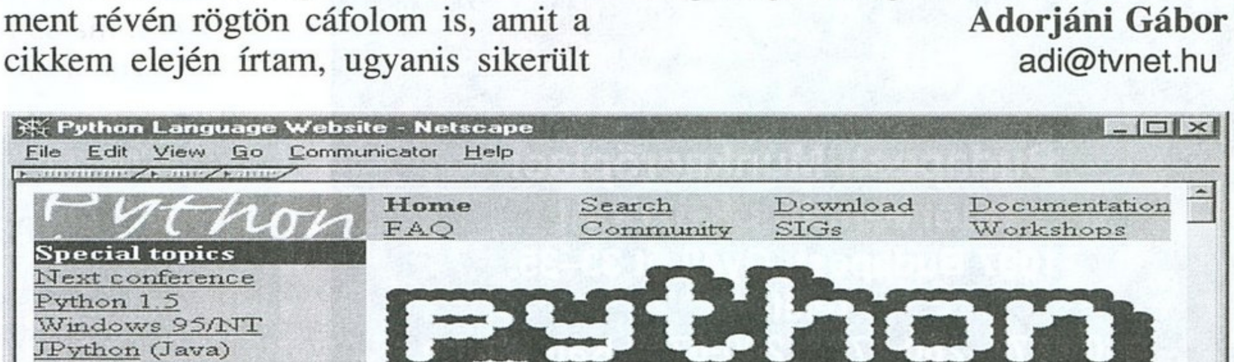
Idézem a dokumentációját: „A Modula-3 rendszerprogramozási nyelv a Mesa, Modula-2, Modula-2+, Cedar nyelvekből lett kifejlesztve, hasonlít az Object Pascalra, az Euclidre és az Oberonra." Minden megtalálható benne, ami egy modern objektumorientált nyelvhez kell, különlegessége, hogy programszálakat (thread) is képes kezelni. Aki használt már Pascalt, az valóban ismerősnek fogja találni a szintaxisát. A csomag egy fordítóprogramot és egy hibakeresőt tartalmaz.

Smalltalk

Tipikus „UFO-jellegű" nyelv, mindenki hallott már róla, de még senki nem látta a saját szemével. A program a Smalltalk-80-nak egy freeware (GNU) implementációja, a bájtkód-értelmező azzal majdnem teljesen kompatibilis. A csomaghoz a CD-n mellékeltem egy segédletet is, hogy az is ki tudja próbálni, aki eddig még nem találkozott vele (én is így voltam vele). Érdekes nyelv, egy futó ismerkedést mindenképpen megérdemel.

Remélem, hogy a CD-re felrakott gazdag programarzenálban mindenki megtalálja a maga kedvencét.

Adorjáni Gábor
adi@tvnet.hu



Rövid hírek az OS/2 világából

Új OS/2 verzió egy éven belül

A PC Week (http://www.zdnet.com/pcweek) hálózati operációs rendszerekről szóló cikkében, IBM-es forrásra hivatkozva tette közzé, hogy az OS/2 Warp következő verziója 1999 első negyedévére várható. A legnagyobb újításként a HPFS386-ot felváltó, az AIX operációs rendszerben már megtalálható Journaling File Systemet (JFS) említették. Ennek a változtatásnak az a lényege, hogy a fájlrendszer naplót vezet a tranzakciókról, és egy esetleges rendszerösszeomlás után a belső adminisztráció alapján másodpercek alatt rendbe lehet tenni a fájlrendszert. Egyúttal megoldják azt is, hogy a partíciók több fizikai merevlemezre is kiterjedhessenek. Az még egyelőre nem világos, hogy a HPFS386 fájlrendszert teljesen kihagyják-e. Ha viszont benne lesz a csomagban, azért valószínűleg felárat fognak kérni, mivel a kód használatáért az IBM is fizet a Microsoftnak. További újdonság még, hogy meg fog szűnni a kliens és a szerver felülete közötti különbség (vagyis a Warp Server megkapja a Warp 4 továbbfejlesztett felületét), megoldják az NT szerverek Warp Serverről történő teljes körű adminisztrálhatóságát, és számos, elsősorban a Workspace On-Demand felhasználói számára hasznos funkcióbővítést is bevezetnek (például BOOTP- és IP-alapú protokollok támogatása).

Távirányítás Java nyelven

1991-ben lépett a számítógépek távirányítására szolgáló programok piacára a svéd Ridax programfejlesztő cég. OS2You névre keresztelt termékükkel szinte bármilyen hálózaton keresztül megoldható volt az OS/2-es PC-k karakteres szekcióinak irányítása. Az OS2You-t rövidesen követte a PM2You, amellyel már a grafikus felület, a Munkaasztal is távirányítható volt. Tavaly kibocsátották termékük Win32-es változatát (WIN2YOU), az idén pedig megjelent a Java nyelven írt kliens is. Ez pedig azt jelenti, hogy a Win95-öt, NT-t vagy OS/2-t futtató gépeket minden olyan munkaállomásról távirányíthatjuk, ahol egy Java 1.1-gyel kompatibilis böngészőprogram van telepítve. A kliensoldalon ehhez nem is kell semmit telepíteni, mivel a böngésző a távirányított gépen futtatott PM2You/WIN2You kiszolgálóról tölti le a szükséges kódot. A program próbaváltozata letölthető a http://www.aditom.se/ridax címről. Hasonló programot fejleszt az International Software Solutions cég is, melynek rövidesen induló bétateszteléséhez vállalkozó szellemű felhasználókat keresnek. Feliratkozni a http://www.iss2you.com/java oldalon lehetséges. Szintén Java nyelven fejleszti Darrel Riekhof a szervomotorok soros vonalon keresztül történő vezérlésére alkalmas programját. Az egyelőre még szintén csak bétának minősülő változat a http://www.busprod.com/ferrettronics/software.html oldalon található.

Ida Pro 3.74 kódfejtő

A DataRescue kiadásában megjelent az OS/2-es változattal is rendelkező interaktív disassembler program újabb kiadása. A leírás szerint az Ida Pro a beépített Flirt technológia révén (http://www.datarescue.com/flirt.htm) felismeri a fordítóprogram könyvtárhívásait ,és ezáltal rendkívüli mértékben felgyorsul a kódvisszafejtés sebessége. Az új verzió már a Pascal és a Delphi fejlesztőkörnyezeteket is támogatja. Az Ida Pro 3.5-tel rendelkezők ingyen juthatnak a 3.74-hez, másoknak 199 $-t kell a programért fizetniük. A program weblapjának címe: http://www.datarescue.com/ida.htm.

Digitális kamerák kezelése

Az egyre nagyobb népszerűségnek örvendő (és egyre inkább elérhető árú) digitális kamerák OS/2 alatti használatát teszi lehetővé Stephane Charette DCITU (Digital Camera Image Transfer Utility) névre keresztelt terméke. A DCITU a következő cégek kameráinak nagy részét már támogatja: Kodak, Agfa, Epson, Olympus, Sanyo, Sierra, Toshiba. További információ a http://members.tripod.com/~dc210/index.html oldalon található.

Előjegyzés a Lotus SmartSuite-ra

A J3 Computer Technologies (http://www.os2store.com) megkezdte az előjegyzések gyűjtését a Lotus SmartSuite új, OS/2-es verziójára. A játékszabályok szerint szinte minden komolyabb irodai csomagról át lehet lépni a Lotus egyre jobban tért hódító termékére. A Warp 4 tulajdonosainak lényeges lehet, hogy a Warp 4-gyel adott IBM Works is a kedvezményes csomagok között van (vagyis minden Warp 4 tulajdonos jogosult az upgrade-re)! Az egyetemi és kutatói szférában dolgozók ezenkívül további árkedvezményben is részesülnek. A SmartSuite teljes ára 379 $, az upgrade ár 139 $, míg az akadémiai ár 79 $. A J3 Computer Technologies külföldi előjegyzéseket is elfogad a sales@os2store.com címen.

XFree86 ablakkezelő

Holger Veit és társai közreműködésével elkészült az XFree86 OS/2-es változatának 3.3.2-es kiadása. Az XFree86 az X consortium X11R6.3-as szabványán alapuló X-ablakkezelő rendszer, melynek OS/2-re hordozott változata az XFree86 for OS/2. A korábbi (3.3.1-es) változattal szemben nyújtott többletszolgáltatás a legújabb videokártyák (ET6100, ViRGE GX/2 és /MX, Rage PRO, I128 Revolution stb.) támogatása és a régebbiek támogatásának tökéletesítése, de több száz hiba javítását is közreadták. A hivatalos XFree for OS/2 honlap címe: http://set.gmd.de/~veit /os2/xf86os2.html.

WebNav 2.0 böngésző

A JMA Software Technologies (http://www.jmast.se/) elkészítette a sokak által az IBM WebExplorer utódjának is tekintett WebNav böngésző újabb változatát. Funkcionalitását tekintve a WebNav nem tud versenyre kelni a Netscape Navigatorral, azonban sebességben és méretben (tömörítve mindössze 320 KB!) messze felülmúlja azt. A cég a fejlesztést a felhasználók igényei alapján végzi, vállalatok számára pedig vállalja egyéni funkciókkal kiegészített böngészők szállítását is. A WebNav DLL formában is elérhető, így más programok is használhatják szolgáltatásait. A WebNav mellett más ismert internetes alkalmazások is megújultak a közelmúltban. Példaként lehet említeni a Binary Newsreader 1.30-as (usenet hírcsoport-olvasó program, http://www.horgen.net/rem/software/) és az Inet.Mail 1.2-es (SMTP/POP3 szerver, http://www.hethmon.com/inetmail.html) kiadásait.

Kádár Zsolt

Magyar Java-decompiler

A Megatrend újdonságaként ismerhetik meg az érdeklődők az Ifabón a világviszonylatban is ritkaságnak számító Java Decompiler Workshopot. (Eddig összesen három ilyen funkciójú, és elég vegyes tudású termék létezett a világ szoftverkínálatában.) Mint ismeretes, a decompilerek olyan programok, amelyek megkísérlik a futtatható programból, a bináris kódból előállítani az eredeti forrást. (Ez leginkább olyan fejlesztéskor hasznos, amikor elkallódott az eredeti forráskód. Tehát nem a jogsértő programfeltörőket hivatott segíteni!) Ami a JDW-ben más, mint a többi decompilerben, az az, hogy a visszafordított program bájtszinten azonos az eredetivel. Emellett minden információ grafikus felületen jelenik meg, továbbá a visszafordítás, újrafordítás és ellenőrzés egy programon belül végezhető el. Egy apróság: ennek a programnak a fejlesztője ugyanaz a Báró Csaba, akinek Clipper decompilerével annak idején az Alaplap hasábjain találkozhattak leghűségesebb olvasóink, s akinek programozási filozófiája azóta sem változott, mert a lefordított decompiler mérete alig 350 kilobájt...

Szövetség a Javaért

IBM, IQSoft, Novell, Oracle, Sun — elég jó kis névsor. A négy multi részvételével — valamilyen variációban — világszerte több alkalmi szövetség is alakult már. (A rossz nyelvek szerint a Microsoft is szerette volna képviselni a maga Java-álláspontját, de képviselőit „eltanácsolták" az asztal mellől.) A magyarországi változatnak külön jelentőséget ad, hogy a közös munka koordinációját éppen a magyar cég végzi majd. Fő célkitűzése a szövetségnek, hogy a Java technológiára és a Corbára alapozott szabványos, platformfüggetlen, komponens-alapú fejlesztési technológia, illetve az erre épülő számos alkalmazás a hazai informatikusok körében váljék ismertté. Ehhez a közös marketinget, illetve a szakmai együttműködést tekintik a leginkább járható útnak. Ugyanakkor a szövetség korántsem jelent szigorú megkötöttséget egyik résztvevő számára sem: mindenki azt adja hozzá a maga szakértelméből, amihez a legjobban ért. A haszonélvező pedig a Java, illetve a felhasználó lehet.

Átalakuló IQSoft

Az IQSoft javás bejelentését egy másik is követte, bár ez elsődlegesen az IQSoft tulajdonosi összetételében bekövetkezett változásokra fókuszált, a változás szakmai tartalma mégis számottevő. Még az elmúlt év végén kikerült a tulajdonosok közül az osztrák banki képviselet (az IQ Management Kft vásárolta ki a részüket), helyére most szakmai befektető került. Az IFS (Industrial and Financial Systems) svéd cég, Magyarországon leginkább ismert termékegyüttesük az Avalon integrált vállalati és vállalatirányítási rendszer, amelyet versenytársai közül kiemel a szabványos, objektumorientált, komponens-alapú technológia. Az IQSoft már korábban is e rendszer disztribútoraként tevékenykedett, a mostani zártkörű tőkeemelés döntő része is e tevékenységi terület további erősödését szolgálja. Az IQSoftnál nagy reményeket fűznek egy további új disztribúcióhoz is: e másik megállapodás az objektumorientált tervezésben, tesztelésben és konfiguráció-menedzsmentben a világpiacon vezető amerikai Rational Corporation termékeire terjed ki.

Lucent-vásárlás a HP-tól

Megerősíti pozícióját a helyhez kötött, vezeték nélküli, szélessávú rendszerek piacán a Lucent Technologies. Ennek értelmében vásárolta fel a Hewlett-Packard LMDS Wireless Business nevű üzletágát (LMDS = local multipoint distribution services — helyi többpontos elosztórendszer). Az LMDS üzleti egység — amely jelenleg a HP videokommunikációs részlegéhez tartozik — vezető szerepet játszik az új generációs mikrohullámú rádiós technológia fejlesztésében. Az új Lucent-divízió Wireless Broadband Networks néven kezdi meg működését.

Nokia monitorok Amerikának

Termelőegységet épít Mexikóban — a Nokia Mobile Phones meglévő gyára mellett — a Nokia Display Products is. Egyértelmű a célkitűzés: a szállítási idő csökkentésével meggyorsítani a termékek amerikai piacra kerülését. Mint ismeretes, a monitor-üzletágnak eddig csak Finnországban és Magyarországon (Pécsett) működnek termelő részlegei.

CeBIT-díj a Novellnek

A CeBIT-en elnyerte az „Innovation of the Year" (Az év újítása) díjat a Novell. A díjra a hálózati kategória zsűrije találta érdemesnek a céget, a nyertes termék pedig az NDS for Windows NT. A hálózati termékek együttműködtethetőségének alkalmából lapunkban is elemzett címtárszolgáltatás közös felületet kínál mind a Novell Netware, mind a Windows NT hálózatok felügyeletéhez.

Synergon—MatávCom

A Lágymányosi híd pesti hídfőjénél álló Lurdy-Ház Bevásárló- és Irodaközpont informatikai és telekommunikációs rendszerét a Synergon—MatávCom konzorcium telepítheti. A feladat különlegessége az épület bérlőinek sokféleségéből adódik. Egyaránt előfordul közöttük 10-20 négyzetméteren árusító apró üzlet, és több száz négyzetméteren működő szupermarket. Mivel az igények folyamatos változására is fel kell készülni, az informatika világában is újdonságnak számító virtuális hálózat, illetve a Synergon Intelligens Épületek koncepciója együtt jelentheti a megfelelő megoldást.

NetFax

Előzetes számítások szerint mintegy 30%-kal csökkenhet a nemzetközi faxok költsége az 1998. április elsejétől elérhető NetFax szolgáltatás révén. Az Euroweb és az amerikai Gric Communications együttműködésének eredményeként született NetFax lényege, hogy a felhasználó a továbbítani kívánt anyagot faxkészülékéről az Euroweb szerverére küldi, ahonnan az — kódolt formában — az Interneten keresztül automatikusan eljut a célországban található hasonló faxszerverre. Innen helyi hívás díjáért jut el, hagyományos telefonvonalon a célállomásra. Elsősorban nagy nemzetközi forgalmat lebonyolító cégek számára jelenthet számottevő költségmegtakarítást. A felhasználó oldalán nem igényel beruházást — még Internet-kapcsolatot sem.

Mobil Pentium II

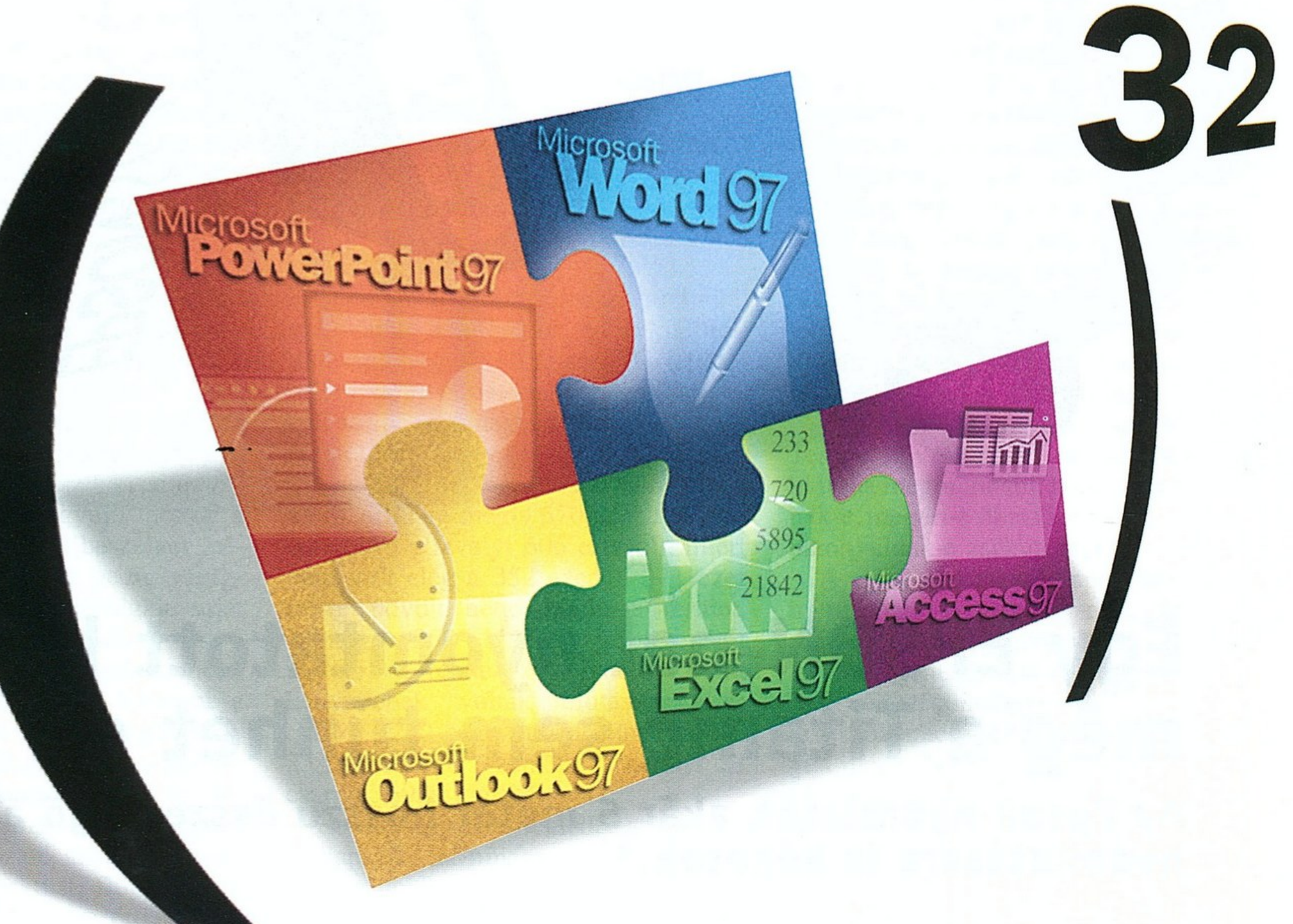
Április elején mutatta be az Intel a hordozható PC-khez kifejlesztett első Pentium II processzorokatt. A 233 és 266 MHz órajelű processzorokat úgy alakították ki, hogy azok rendelkezzenek a P6 mikroarchitektúra korszerű teljesítményjellemzőivel, ugyanakkor feleljenek meg a hordozható számítógépes környezet különleges energiafogyasztási és méretigényeinek. A könnyítést egy újszerű minikazetta tette lehetővé, amely magát a processzort és a második szintű cache-t tartalmazza: négyszer könnyebb, hatszor kisebb — és feleannyi energia fogyasztását követeli meg, mint az asztali gépek Pentium II-inek SEC kazettája.

Végleges Quicktime

Elkészült az Apple szabványként elfogadott Quicktime 3.0 multimédia-architektúrájának végleges változata. Első tapasztalataink alapján a rendszerrel kapcsolatban nem osztjuk ugyan a szabványteremtők optimizmusát, de ismerkedésül CD-mellékletünkön közreadjuk az Internetről is letölthető szoftverváltozatot. Az extra szolgáltatásokkal és kiegészítőkkel felvértezett, megvásárolható Quicktime 3.0 Pro valószínűleg többet tud...

Hatékonyság, stabilitás, könnyű kezelhetőség, szervezettség

...mostantól már a harminckettediken!



Részletes tájékoztatásért hívja a 2MSINFO (267 4636) telefonszámot! http//www.microsoft.com/hun/32

A számítógép használhatóságát több tényező együttese határozza meg. Egy gyenge minőségű vagy illegális szoftver az egész gép értékét csökkentheti. Különösképpen igaz ez az üzleti alkalmazásokra, melyek nagymértékben meghatározzák az iroda munkájának minőségét.
Válassza Ön is a magyar nyelvű 32 bites Microsoft® Office 97 Professional-t! – meglátja: e szoftvercsomaggal a vállalata mérhetően hatékonyabbá válik, ...és még a kollégái is elégedettebbek lesznek.

SCRIPTUM

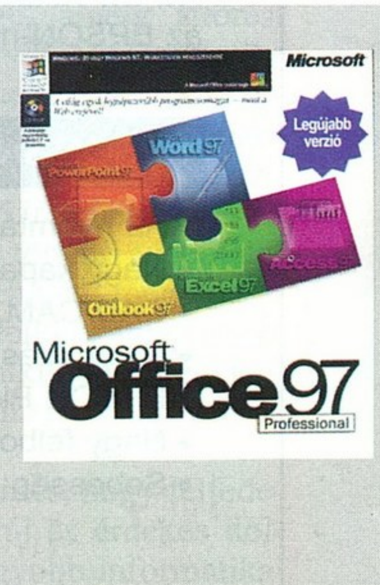
A Microsoft 32 napig tartó ajánlata



- Vásároljon magyar nyelvű, dobozos, 32 bites Microsoft® Office 97 Professional-t és egy **Scriptum szótárcsomagot* kap ajándékba.**
- Amennyiben legalább 5 db Microsoft® Office 97 Professional-t** vásárol, a kedvezményes ár mellett egy **Scriptum szótárcsomagot*** és egy **Timex Data Link** karórát ajándékozunk Önnek; az óra képes adatokat beolvasni – és tárolni – a számítógép képernyőjén keresztül.
- Ha Ön legalább 10 db Microsoft® Office 97 Professional-t** vásárol, egy **Hewlett Packard 320LX Palmtop**-ot kap ajándékba.



* A szótárcsomag az Akadémiai Kiadó és a Scriptum Kiadó szótárait tartalmazza: Angol-magyar, magyar-angol hangosszótár, Helyesírási kéziszótár, Helyesírási szabályzat, Idegen szavak szótára, *Scriptum*: Angol-magyar informatikai szótár.
** Az ajánlat Microsoft Open Licence Pack vásárlása esetén érvényes.
Az ajánlat 1998. május 4-től június 4-ig tart.



Microsoft



Egy EPSON-nal nyomtatott képről még a Titanic sem tűnhet el.

Az Epson nyomtatók akár 5 méter hosszú összefüggő ábra nyomtatására is képesek.*

Ha Ön vállalkozása, grafikai stúdiója számára keres megbízható, professzionális nyomtatót, az EPSON-nak rögtön három ajánlata is van:

EPSON *Stylus* COLOR 1520

- A2 nyomtatási méret
- Nagy kapacitású tintapatron
- CAD/CAM, gyorslistázás
- Nyomtatás tekercspapírra 5 m-ig
- EPSON Plot alapkiépítésben
- Nagy felbontás – 1440 dpi
- Sebesség: 8 lap/perc

EPSON *Stylus* COLOR 3000

- Teljes A2 nyomtatási méret
- Nagy kapacitású tintapatronok
- CAD/CAM, gyorslistázás
- Nyomtatás tekercspapírra 5 m-ig
- Nagy felbontás – 1440 dpi
- Sebesség: 8 lap/perc

EPSON *Stylus* Pro 5000

- A3+ nyomtatási méret, 6 szín
- Nagy kapacitású tintapatronok
- PROOF nyomtatási lehetőség
- CAD/CAM, gyorslistázás
- Nagy felbontás – 1440 dpi
- Sebesség: 8 lap/perc

Kizárólagos importőr TRADE R.A. Trade Kft.
Fax: 06 (23) 417 310
EPSON HOTLINE: 06 (23) 415 295

*EPSON® *Stylus*™ COLOR 5000-re nem vonatkozik

EPSON®

INFORMÁCIÓKÉRÉS: 32 ◄

Tavaszi „zarándoklat” Hannoverbe

Láttál valami érdekeset is?

A vásárlátogatásról hazatérőket gyakran fogadják kollégáik és ismerőseik a címbeli kérdéssel. Miután azonban mindenki más szemmel nézi a dolgokat, nagyon különböző, sőt egymással ellentétes véleményeket is hallhatunk ugyanarról a rendezvényről. Különösen, ha az akkora és olyan sokarcú, amilyen az idei CeBIT volt. Hogy a mennyiség mennyiben jelent minőséget, arra az újdonságok felől próbálok az alábbiakban — kissé talán elvontnak tűnő — választ adni, de a Paletta rovatban (a 39. oldalon) bemutatok hozzá néhány egészen konkrét illusztrációt is.

Egy szűkebb területre szakosodott szakember vásárlátogatása tipikus esetben úgy alakul, hogy egy vagy két alkalommal kimegy a „terepre”, megnézi az őt közvetlenül érdeklő és általa jól ismert témákban bemutatott technikai újdonságokat, igyekszik kiszimatolni azok részleteit, begyűjti hozzá a fellelhető anyagokat, elbeszélget néhány partnerével, utólag pedig végiggondolja a látottak tanulságait saját maga és cége szempontjából.

Egy szakmai újságíró ezzel szemben igyekszik minél többször kimenni a vásárra, fizikai erőnlététől függően végiggyalogol lehetőleg az összes pavilonon és az összes stand mellett, felületesen végignéz mindenfélét, begyűjt sok kilónyi tájékozató anyagot (sajtóközleményt, prospektust, floppyt meg CD-t) az olvasói számára érdekesnek tartott újdonságokról, igyekszik a cégek képviselőivel személyes beszélgetést folytatni a különösen izgalmas témákról, elmegy néhány sajtótájékoztatóra (abból a mintegy 220-ból, amely az idei CeBIT első napjaiban lezajlott), majd pedig a vásáron szerzett tapasztalatokból, élményekből, információkból legalább fél évig közvetve vagy közvetlenül, tudatosan vagy öntudatlanul sok mindent felhasznál a lapba kerülő témákhoz, anyagokhoz.

Én a második kategóriába tartozom, és talán ebből fakad, hogy ezekről a hatalmas seregszemlékről visszatérve utólag sem alakul ki bennem negatív összkép. A legrosszabb esetben is gazdagodom egy sor hasznosítható új információval és rengeteg szakmai élménnyel, miközben ugyanarról a rendezvényről más lapokban azt olvasom, hogy a vásáron „nem volt igazán nagy durranás” ... „elmaradtak a szenzációs bejelentések” ... „amiket láttunk, azokról már jóval előbb mindent tudtunk” stb. Vajon a látogatók és a kiállítók többsége hogyan éli át ezt? Ha feltételezem, hogy én látom tévesen a dolgokat, akkor nem nagyon találok magyarázatot arra, hogy (az előző évek „csalódásaiból” okulva) miért ment el az idei CeBIT-re mégis 670 ezer látogató (szemben a tavalyi 606 ezerrel), miért nem volt elég a 352 ezerről 371 ezer m^2-re bővölt nettó kiállítási terület. Mintegy 250 kiállító jelentkezését ugyanis a rendezők helyhiány miatt kénytelenek voltak elutasítani, és „csak” 7250 céget tudtak elhelyezni a 25 csarnokban (szemben a tavalyi 23 csarnok 6909 kiállítójával).

Lehet, hogy az érdeklődés akkor sem csillapszik, ha nincsenek igazán nagy újdonságok? Kétségtelen, hogy a szakmai újságírás helyzete ebből a szempontból kicsit más. Témaként egy technikai újdonságnak a gondolati fogantatástól a fizikai megszületésig terjedő időszaka a legizgalmasabb. Ezt követően érdekes még az első termékek útkeresése és a gyermekbetegségek korszaka is, utána azonban, a kiforrott szériatermelésben megtestesülő piaci életciklusban az intellektuális kalandra vágyó szakemberek képzeletét már nem nagyon lehet a megszokott dolgokkal megragadni.

És itt lép be a vásári beszámolóknak azon kockázata, hogy a látottak alapján nem mindig lehet tartós trendekre következtetni. Néhány éve a CeBIT egyik nagy slágere, látványos újdonsága a Momenta és a vele egyidejűleg kifejlesztett többi kézírásos adatbeviteli rendszer volt, elektronikus tollal, interaktív képernyővel stb. Ez a „vágytalálmány” azonban nem élte túl a csecsemőkorát, a vásári kínálatból mostanra teljesen eltűnt, és néhány hete még az Apple is bejelentette, hogy abbahagyja a legtovább kitartó, zsebbe való „kistestvér”, a Newton fejlesztését.

Az idei CeBIT-en számos olyan újdonságot mutattak be, amely nem puszta számfétisizmus (több megabájt, több megahertz és hasonlók), hanem lényeges technológiai irányváltás. Ezek életképességének megítélésekor azonban lehet, hogy ismét csak a vágyainkat fejezzük ki. Hiszen persze, hogy szeretnénk, ha számítógépünket hanggal tudnánk vezérelni, vagy ha a magnóra rögzített beszédet szemvillantásnyi idő alatt és gépírónő közbeiktatása nélkül bevihetnénk a szövegszerkesztőnkbe. Tessék, itt vannak rá a megoldások. De ki tudja megmondani, hogy ezek beválnak-e, megéri-e őket kipróbálni, és nem bizonyulnak-e zsákutcának?

A falra akasztható lapos képernyő iránti vágy is nagyon régi. Most a számítógép mellé helytakarékos monitorok új generációját fejlesztették ki a TFT LCD technológiával, és lenyűgöző kivetítőket készítettek plazmatechnológiai megoldással. A szakmai látogatók már nem azt kérdezték, hogy mennyire lehet még javítani a falsíkhoz illeszkedő megjelenítők képminőségét, hanem hogy mennyire lehet „javítani” az egyelőre elég magas árat.

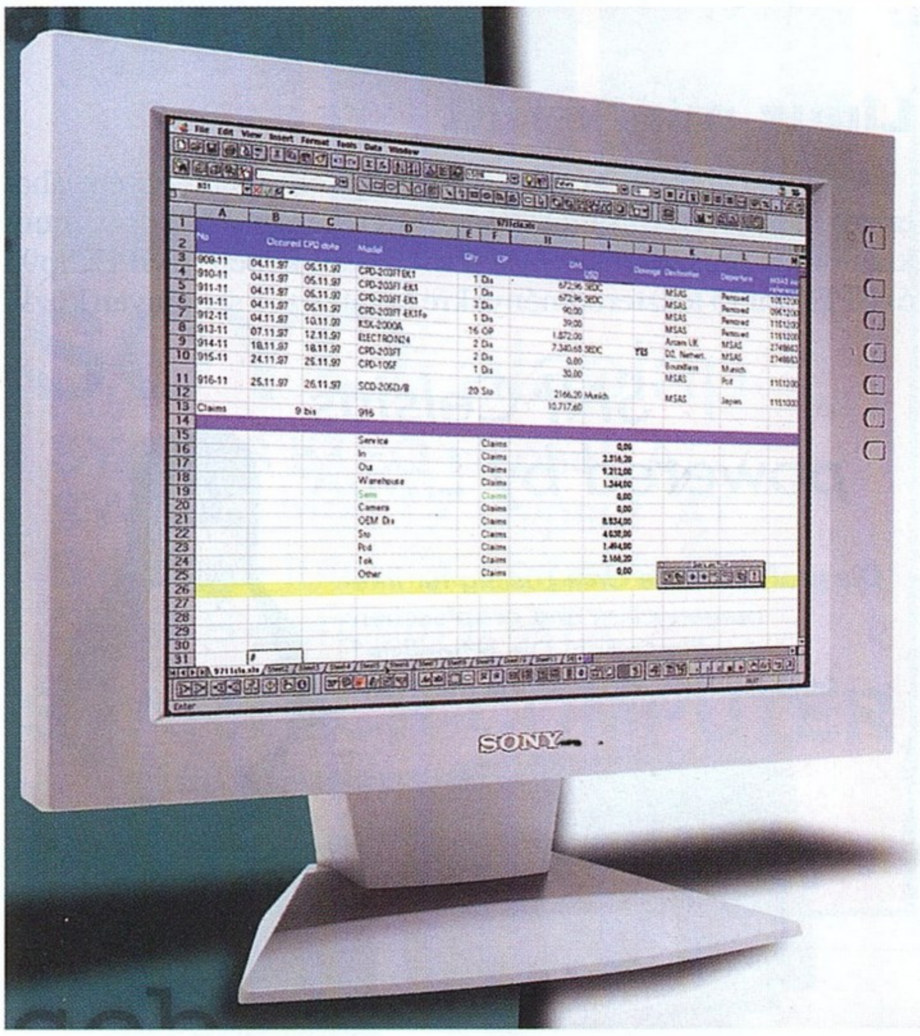
Ugyanígy sok egyéb témában is bőven volt igazi érdekesség: a digitális kamerák sokfélesége, amely áttörést hozhat a fotózás és a videózás területén; a DVD írható változatának közös szabványa, melyet sajtótájékoztatón mutattak be a nagyvilágnak; a tévécsatornákon keresztül történő internetezés és az Interneten keresztül történő videotelefonálás, melynek technikai megoldásai már lépten-nyomon felbukkantak, csakúgy, mint az elektronikus kereskedelem lebonyolításához szükséges hardver- és szoftvereszközök...

Nem is az a fő kérdés, hogy felfedezünk-e magunknak új és érdekes dolgokat a világ legnagyobb informatikai rendezvényén (hiszen ilyesmit mindig rengeteget találhatunk), hanem hogy mi lesz az újdonságok jövője, s hogy jó lóra teszünk-e, fejlesztőként, kereskedőként vagy az innovációk iránt fogékony vevőként...

Faklen Pál

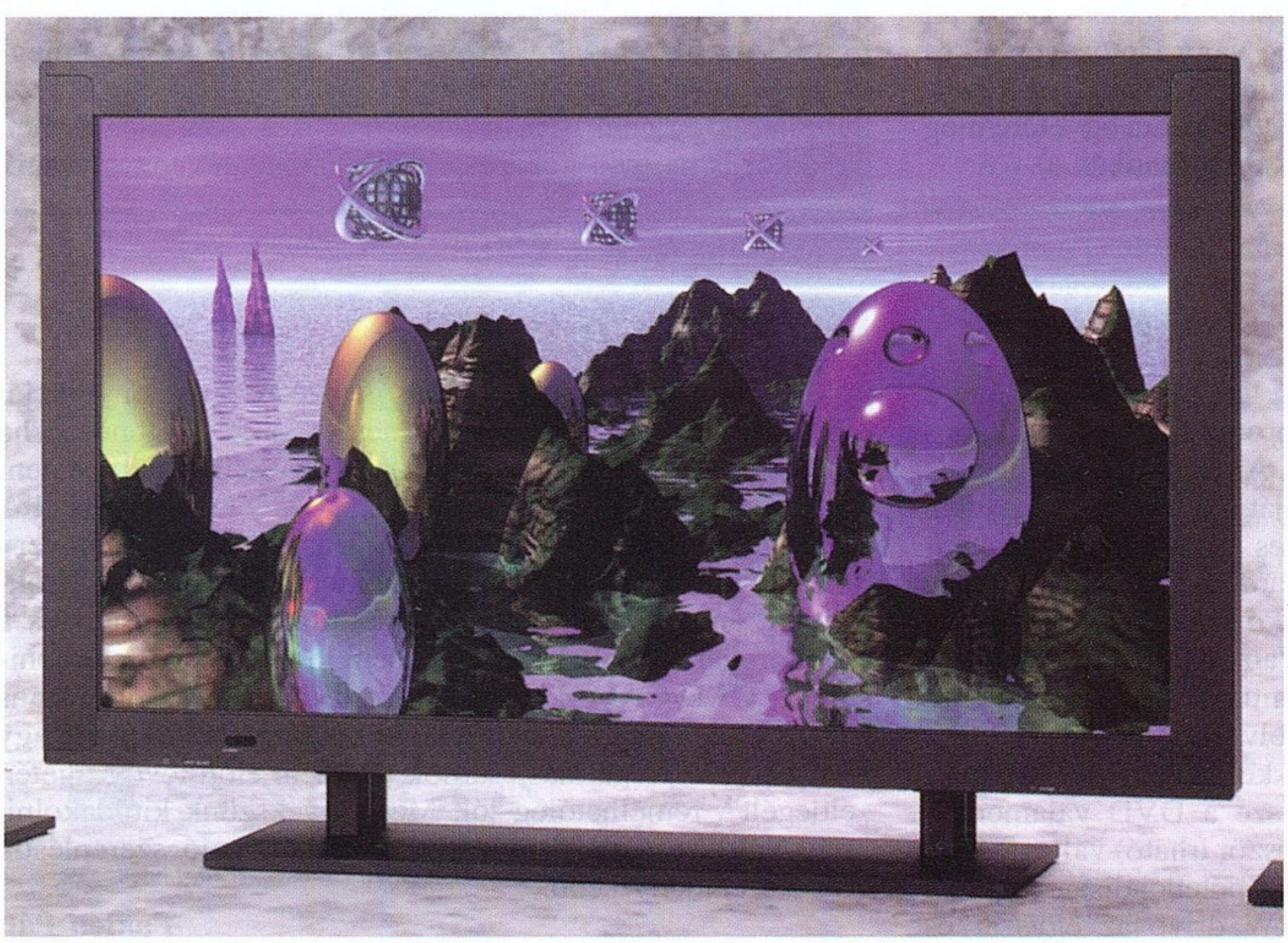
A kívánatos ellaposodás

A március 19-25. között lezajlott CeBIT '98 legnagyobb élményét számomra a lapos képernyők sokasága jelentette. A számítógépmonitorok közül véletlenszerűen a Sony L150 típusjelű, 15 collos képátlójú készülékét választottam a TFT



LCD technológia képességeinek illusztrálására. A vékonyrétegű aktív tranzisztorcellák (TFT = thin film transistor) és a polarizált folyadékkristályos képelőállítás (LCD = liquid crystal display) kombinálásával sikerült kiküszöbölni az eddigi LCD-k legnagyobb hibáját, a keskeny rálátási szöget. Itt a vízszintes irányú rálátási mező már 140 fok! A torzításmentesség, a színhűség, a képfelbontás, a képélesség, az erős kontraszt, a külső fényviszonyokhoz alkalmazkodó fényerő és a sugárzásmentesség révén a TFT LCD alkalmas lehet a katódsugaras technika teljes kiszorítására is (bár ez a megállapítás kissé jóslat ízű, ezért lásd erről a 37. oldali cikket is). Az L150 monitor súlya 5,3 kg, teljes vastagsága 16,5 cm. Nem sorolom egyéb paramétereit, mert mindent tud, ami ma egy monitortól elvárható. Látni kell.

A nagyméretű képernyők világát a jelek szerint a plazmatechnológia fogja átrendezni. Az új képernyő a fluerszkáló fény elve alapján működik. Minden egyes képpont egyben saját fényforrás is, három színcellával (RGB), melyek mindegyike külön elektróddal vezérelhető. Az elektródra érkező elektromos töltés hatására a cellába zárt gáz ultraibolya fényt bocsát ki, és a cellánként különböző (vörös, zöld és kék) foszforréteget a megfelelő látható szín kibocsátására gerjeszti. A cellák szigetelésére igen vékony (100-200 mikronos) üvegréteg szolgál. Egyszerre több cég is előállt az újdonsággal (Hitachi, NEC, Pioneer, Sony), de szubjektíve nekem a „szélesvásznú" NEC PlasmaSync 4200W tetszett legjobban. Ez a képoldalak normál 4:3 arányát 16:9 arányúra tudja módosítani, ami persze inkább videomód esetén érvényesül. A képernyő 92 x 52 cm méretű (42" az átló), felbontása 853 x 480 pixel, analóg RGB módban 16,7 millió színt produkál. Állvány nélkül a képernyő vastagsága mindössze 8,9 cm, súlya 39 kg. A kép teljesen torzításmentes, és még egy apróság: a rálátási szög 160 fok. Ezt is látni kell.



Diktáld a szöveget gépbe

A számítógéppel szóban közölt parancsokat régóta „megoldották" a tudományos-fantasztikus filmekben, de a számítástechnikai kivitelezés lassabban megy. A fejlesztők úgy látták, hogy jóval nagyobb gyakorlati jelentőségű lenne, ha normál szövegeket lehetne a számítógépnek lediktálni és azok rögtön úgy jelennének meg, mintha egy szövegszerkesztőbe lettek volna begépelve. Egyrészt mert életünk sokkal inkább tele van szöveges anyagokkal, mint parancsokkal, másrészt a sima szövegek elhallásai nem okoznak nagyobb gondot, mint a gépírónők elütései, szemben azzal, hogy bizonyos helyzetekben egyetlen parancs félrehallása is súlyos következményekkel járhat.

A beszédfelismerés úttörő cége, a Dragon Systems már 1984-ben kirukkolt az első ilyen rendszerrel, és azóta nem csekély kutató- és fejlesztőmunkával sikerült egyre jobban finomítaniuk a beszéd szöveggé alakításának szoftveres megoldásait. A legújabb Dragon Naturally Speaking hibaszázaléka ma már 5% alatt van. Be kell vallanom, elég meggyőző dolog látni, ahogy a mikrofonba folyamatosan diktált szöveg a képernyőn szinte egyidejűleg jelenik meg



„beírva" a szövegszerkesztőbe, sőt utasítást adhatunk (lám, lám... mégiscsak) szövegrészek kiemelésére (betűtípusra, kövér- vagy dőltbetűs szedésre, középre igazításra stb.), és lebetűzhetjük a nehezen érthető szavakat. Az így keletkezett szöveg azonnal szerkeszthető pl. WinWordben vagy más windowsos szövegszerkesztőben. Diktáláskor rögzítésre kerül a hangfelvétel is, tehát utólag bármelyik részlet visszahallgatható. Emellett persze egyéb írott szövegeket is felolvastathatunk a rendszerrel.

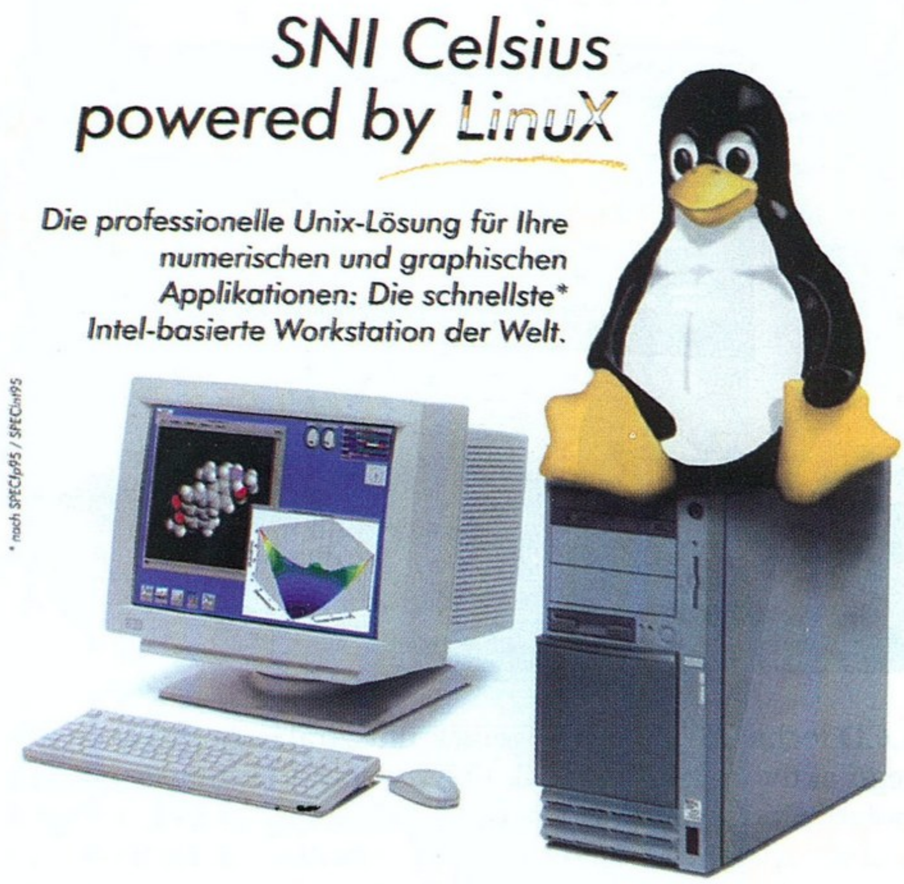
A beszédfelismerési technológia régóta foglalkoztatta az IBM kutatóit is. Kifejlesztették a Simply Speaking és a ViaVoice szoftvereket, majd ezek kibővített (Gold) változatait. Most a CeBIT-en megjelent az ugyancsak IBM technológiára alpozott VoiceOffice 98 (a Linguatec terméke), mely hasonló tudású, mint a Dragon Naturally Speaking szoftver. Ebben a műfajban legjobban mégis az Olympus D1000 digitális zsebdiktafon keltette fel érdeklődésemet (amely egyébként szintén az IBM ViaVoice technológiáját alkalmazza). A mellékelt képen is látható 12,1x4,6x2,3 cm-es készülék 2, illetve 4 MB kapacitású flash memóriakártyával 33, illetve 70 perces beszéd rögzítését teszi lehetővé. A diktafon flashmemóriájának tartalma egy összekötő kábelen keresztül pillanatok alatt letölthető egy PC-be vagy laptop gépbe, és ott szöveggé alakítható. A digitális működésmódból következik számos egyéb szolgáltatás is... Ja, az ára egyelőre még jóval meghaladja a „hagyományos" diktafonokét.

DVD+RW — a jövő meghajtója

Az idei CeBIT-re időzített egyik legnagyobb esemény a DVD+RW drive bejelentése és a prototípus első nyilvános demonstrációja volt. Elég nagy a tét. Az asztali számítógépek jelenlegi floppy- és CD-meghajtóját helyettesítő egyetlen univerzális meghajtóra kell egységes szabványokat kialakítani. A DVD+RW a Hewlett-Packard, a Philips és a Sony közös fejlesztése volt, ehhez csatlakozott még három japán cég, a Mitsubishi, a Ricoh és a Yamaha. Az új meghajtó „mindenevő" és majdnem „mindentudó", ami azt jelenti, hogy megvan a visszamenőleges (olvasási) kompatibilitása a csak olvasható CD-ROM lemezekkel, az egyszer írható és a többször írható CD-kkel, és persze a DVD valamennyi (csak olvasható, egyszer írható, többször írható) változatával, az írást illetően viszont egyelőre csak potenciálisan van meg a lehetőség arra, hogy a DVD+RW drive az egyszer és többször írható CD-kre is írjon. Az alkalmi szövetséget alkotó 6 cég együttvéve 75 százalékát uralja a magneto-optikai és a CD-írási technológiának, így talán van remény arra, hogy ezen a területen nem tör ki újabb szabványháború. (Balhé lesz persze enélkül is, lásd legutóbbi számunk 32. oldalán a világ területi újrafelosztását.)

Linux mindenütt

A Linux-pingvin képére figyeltem fel olyan helyen, ahol erre nem nagyon számítottam. Egy Siemens Nixdorf munkaállomás prospektusa (lásd a mellékelt képen) a SPECfp95 és SPECint95 teszteredményekre hivatkozva szerényen közli,



hogy ez a világ leggyorsabb olyan munkaállomása, amely Intel processzorral működik, mégpedig azért, mert a numerikus és grafikus alkalmazásokra professzionális Linux-megoldást dolgoztak ki. Lehet, hogy jön majd valaki, aki azt mondja, hogy nem is ez a leggyorsabb, hanem a ... Lehet, de akkor is nagyon jó érzés az alternatívák sikerét olvasni!

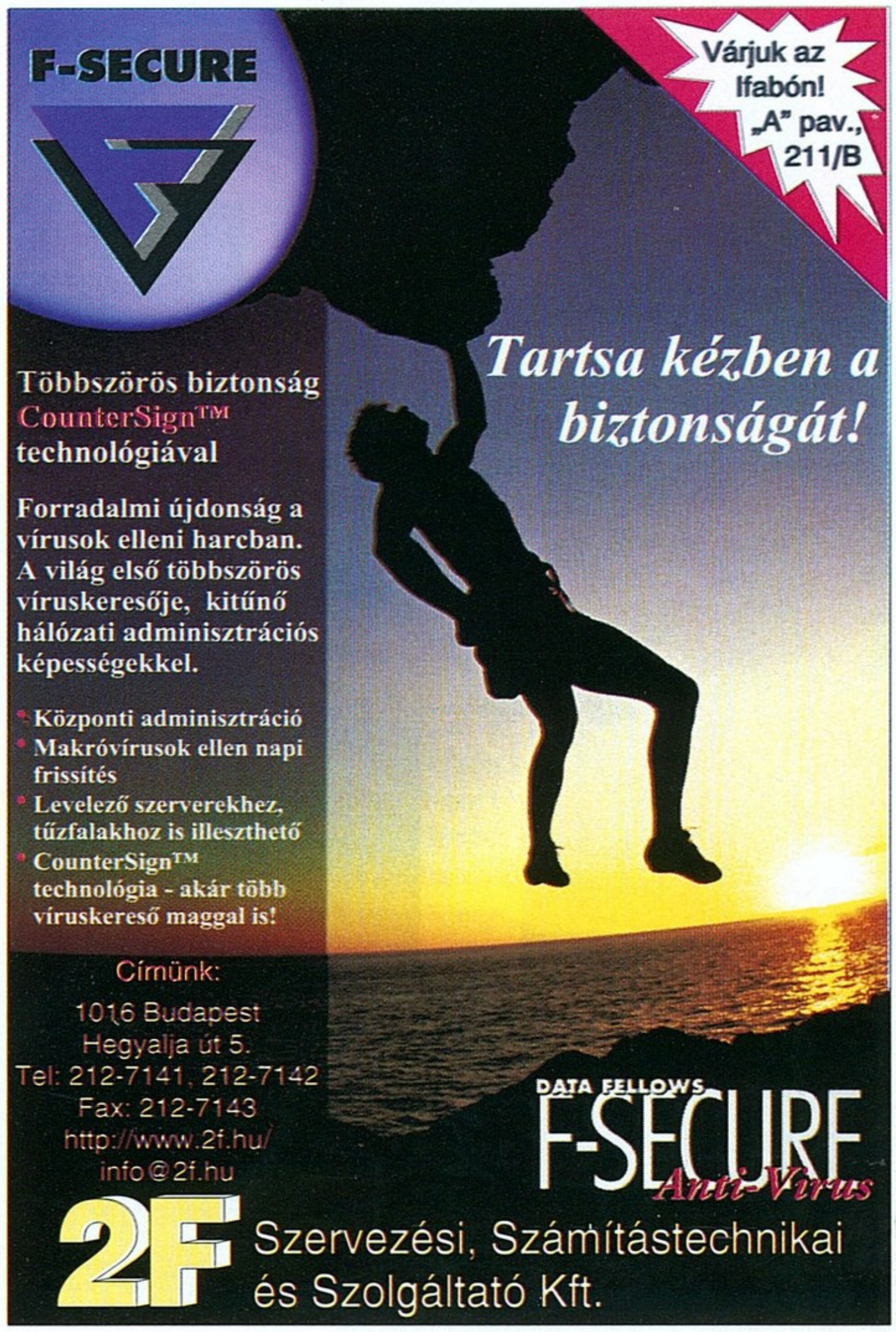
Hajrá, magyarok!

Magyar kiállítóknak ilyen erős nemzetközi mezőnyben jól észrevehetően megjelenni soha nem könnyű, különösen ha a stand elhelyezése is előnytelen, ami az utóbbi években nem egyszer előfordult. A mostaniak kaptak egy kis kárpótlást! A Kürt Kft (nemzetközi nevén EDRS, European Data Recovery Service) olyan csarnok főútvonala mentén kapott egy sarokstandot, amelyben a domináns kiállítók a winchester- és a monitorgyártók voltak. Kívánni sem lehet jobbat! A Graphisoft a CAD/CAM kiállítók között ugyancsak impozáns helyen volt, a Recognita és a többi magyar kiállító pedig egy nem kevésbé forgalmas csarnok jól szem előtt lévő szigetén „éldegélt", remélhetőleg jól. Majd igyekszünk kitudakolni, hogy vajon a CeBIT-en vagy az Ifabón való szereplésük volt-e eredményesebb.

Faklen Pál



INFORMÁCIÓKÉRÉS: 36 ▶



INFORMÁCIÓKÉRÉS: 01 ▶



a RENDSZERINTEGRÁCIÓ FILOZÓFIA.
Megálmodni
Megvalósítani
MŰKÖDTETNI

A rendszerintegráció több, mint hardver, több, mint szoftver. A rendszerintegráció szemléletmód: az informatika filozófiája. A Synergon, a piacvezető hazai rendszerintegrátor egyesíti az informatikát és a kommunikációs technológiát, így hatékonyságot növelő, költségcsökkentő rendszereket hoz létre kiemelkedő minőségű alkotóelemekből.

Az informatikai rendszerintegráció több, mint rendszer, mert nemcsak tervezést, építést, felszerelést jelent, hanem azt a hátteret is, amely a rendszerek gazdaságos működtetéséhez, a piaci sikerhez elengedhetetlenül szükséges. Ha ismeri a célt, a Synergon megépíti a hozzá vezető utat!

SYNERGON

ISO 9001 SGS

Synergon Informatika Rt. ■ 1041 Budapest, István út 16. ■ 1047 Budapest, Baross u. 91-95. ■ Tel.: 399-6600 Fax: 399-6699 ■ E-mail: info@synergon.hu www.synergon.hu

INFORMÁCIÓKÉRÉS: 41 ▶


IFABO
BUDAPEST
1998
Május 5-9.
Nemzetközi számítástechnikai,
kommunikációtechnikai és
irodaszervezési szakvásár
Budapesti Vásárközpont
Nyitva tartás: hétköznap 10–18 óráig
szombaton 10–16 óráig
INFORMÁCIÓKÉRÉS: 20 ▼

Set-top

A WinHEC fórumon Windows CE-alapú set-top box referenciagépet mutatott be az ATI. A Set-top-Wonder CE magában foglal egy gyors x86-os processzort, egx 56 kbites modemet, egy SoundMAX 64 hangkártyát és egy Rage Pro videokártyát, ImpacTV2 csatolóval és tévétunerrel együtt. Ez jelenleg az egyetlen set-top készülék, amelyik képes a DVD és az MPEG-2 lejátszására.

Video

A C-Cube DVD-interfészt hozott ki a nemrég megjelent Intel I740 grafikus kártyákhoz. A Chelsea-I alapja a ZiVA chip, az első olyan hardveres DVD-megoldás, amely közvetlenül csatolható az Intel740-hez. A DVD-kiegészítő kártya már rendelhető OEM-eknek közvetlenül a gyártótól vagy az Inteltől.

Az nVidia bejelentette következő nagy teljesítményű 3D chipjét. A Riva TNT névre keresztelt változat kitöltési sebessége 250 millió pixel, ez eddig a legnagyobb, amit az 1 chipes eszközöknél megadtak (a Voodoo2 kétchipes) — ezt úgy éri el, hogy egy órajel alatt két képpontot számít ki —, és akár 8 millió háromszög/s teljesítményre is képes. Ezenkívül AGP 2x-es módban is működik, 16 MB-ig bővíthető, 250 MHz-es RamDacja megfelelő képfrissítést biztosít a legnagyobb felbontás esetén is. Sorozatgyártása 1998 harmadik negyedévében indul be.

Egy felmérés szerint (amely az üzletek 35%-át vizsgálta) 1997-ben a kiegészítő, csak 3D kártyák piacán az USA-ban a Voodoo chipes kártyák 85 százalékot értek el, több mint 6-szor annyit, mint a legközelebbi vetélytárs, a NEC PowerVR-je.

Audio

Az Aureal Vortex chipjét választotta a Turtle Beach a Montego A3DXstream PCI kártyájához. Az Aureal A3D technológiájának köszönhetően valódi térhangzás állítható elő szabványos sztereó hangszórókkal, ezenfelül 64 hangú, stúdió minőségű hullámtábla-szintézis megszólaltatására képes. (Az Aureal megkapta a szabadalmi jogokat a 3D audiotechnológiájára, ezzel hivatalosan is elismerték a módszert.)

Driverek

A Silicon Graphics és az OpenGL Architecture Review Board többi tagja elfogadta az OpenGL 1.2-es új specifikációját. Fejlesztették a 3D megjelenítés minőségét, növelték sebességét, képességeit. A Windows-támogatás mértékét is kibővítették, például közvetlenül támogatja a Windows pixelformátumait. Az új API várhatóan ősz végére jelenik meg a különböző platformokon.

Az ATI elkészítette Rage Pro sorozatának OpenGL meghajtóját Win95 alá. A driver egyelőre tesztfázisban van (béta), de május közepére ígérik a teljes változatot. A jelenlegi állapotban is képes futtatni a népszerű Quake programot, az egyéb 3D alkalmazásokhoz azonban még várni kell a végleges verzióra.

A Cirrus Logic több mint 3 millió ARM-alapú RISC processzort szállított az elmúlt üzleti évben. A Cirrus saját terveivel kiegészített chipeket azokon a helyeken hasznosítják leginkább, ahol az alacsony fogyasztás és a kis méret fontos, például a mobil kommunikációban.

A nálunk Panasonic termékeiről ismert Matsushita vállalat szerződést kötött a Rambus céggel a Rambus technológia alkalmazásáról. E technológia előnye, hogy a sávszélesség növelésével szemben inkább a kommunikációs frekvenciát növelték, így kisebb integrációjú lapok alkalmazásával is elérhető a nagy átviteli sebesség.

A Jazz Multimedia fejlesztői elsőként készítettek olyan grafikus kártyát, amely egyaránt alkalmazható PCI és AGP buszon is, így az összeszerelőknek nem kell azzal foglalkozniuk, hogy az éppen milyen készülékbe kerül. A kártya Rendition V2200 chipje már számos gyártó kártyáján megtalálható.

A legújabb Glint GMX 1000-es chipjét alkalmazza GLoria-XXL nevű AGP buszos CAD-gyorsítókártyáján az Elsa. A kártya 16 MB VRAM megjelenítő és 24-től 40 MB-ig bővíthető DRAM textúramemóriával készül, 3,3 millió sokszög/s a maximális teljesítménye. Felbontása HDTV-kompatibilis, 1920x1080 vagy 1600x1280 pixel is lehet. Az Elsa speciális MAXtreme drivereivel a 3D Studio MAX akár 2,5-szer gyorsabb lehet komplex textúrázott 3D szerkesztések esetén.

A Creative újabb DVD-csomagot dobott piacra. A Creative Components 700 nemcsak a DVD-meghajtót és a lejátszókártyát tartalmazza, hanem egy AWE 64 hangkártyát és egy 4 MB-os, 2 chipes videokártyát, valamint Cambridge SoundWorks' PCWorks hangszórórendszert, azaz gyakorlatilag mindent, ami a multimédiához szükséges. A készlet ára 480 dollár, ami hazai mércével sok, de az önálló DVD-lejátszóknál olcsóbb. A Creative külön DVD-drive-ot is forgalomba hoz, annak várható ára 150 dollár lesz.

A Dinamic Pictures, az Oxigen 3D kártyáival eddig a professzionális felhasználók számára lehetett inkább ismert. Mostani bejelentésük első fele megint a profiknak szól, mert új drivereikkel az OpenGL-támogatás többszálú futtatását teszik lehetővé, ezáltal jelentős gyorsulás érhető el a többprocesszoros rendszereknél a geometriai feldolgozásban. A másik rész viszont az árak csökkentéséről szól, így a legolcsóbb modell, a 8 MB SDRAM-os Oxigen 400 dollár lett, ez már a közepes tervezőrendszerekben eddig alkalmazott kisebb teljesítményű kártyákhoz mérve is kedvező ár.

A NEC is bejelentette PowerVR sorozatának legújabb tagjait, nemcsak PC, hanem játéktermi és konzolgépekhez is. Az új chipek teljesítményben és minőségben többet nyújtanak elődeiknél. A 3D tárgyak átfedéseihez használt 32 bites Z-puffer sokkal nagyobb precizitású megjelenítést tesz lehetővé, mint az eddigi 16 bites változatok.

Szoftver

SoftDVD II néven hozta ki Direct-Show-kompatibilis lejátszóját a Zoran. Ez nagyobb fokú integrációt tesz lehetővé a multimédia-alkalmazásokban, és megkönnyíti a kezelést. Jelenleg a SoftDVD alkalmazói között van a Compaq, a HP és az ATI.

A Microsoft több cégtől is átvesz technológiákat a DirectX 6-os készletéhez. Az S3-tól a textúratömörítő, a Tritechtől a bump mapping eljárást fogja alkalmazni.

Az Imagine Studios „Ares Rising" című játéka egyike lesz azoknak, amelyeket optimalizálnak az AMD K6 3D processzorához. A játék egy űrharcszimulátor lesz, gazdag 3D grafikával, szerteágazó történettel, és több mint 50 küldetéssel.

Per

A Creative Technology és E-mu Systems leányvállalata perli a Diamond Technologiest és az ESS-t, azt állítva, hogy a Maestro-2 audiochip az általuk bejegyzett módszert is alkalmazza a hangkeltésben. Az ESS — miután megvizsgálta a szóban forgó szabadalmat — azt állította, hogy nem követett el jogsértést.

Bánó György

3Com: Új OfficeConnect kapcsolók

OfficeConnect rendszerének olyan további tagjaival jelent meg a piacon a 3Com cég, amelyek a kapcsolók funkcióit és teljesítményét, valamint a nagyobb sebességű LAN-okra való átlépést teszik lehetővé akár a legkisebb irodai környezetben is. Az új OfficeConnect Switch 280 tíz darab 10Base-T és két darab 100Base-TX kapcsolóporttal rendelkezik; az OfficeConnect Switch 400-nak pedig négy darab 10/100Base-TX kapcsolóportja van. Mindkét switch könnyű migrációt kínál a 10 Mbit/s-os hálózatoknál a 100 Mbit/s-osokra. Az új eszközök megjelenésével egyidejűleg, ez év márciusától a 3Com 45 százalékos mértékig terjedően csökkentette egyes, már korábban is meglévő OfficeConnect kapcsolóinak az árát. Így az USA-ban a 140M típus 495 dollárért, SNMP és web-alapú menedzsment-funkciókkal pedig 895 dollárért érhető el. Az OfficeConnect rendszer jelenleg 32 termékből áll. Ezzel — a 3Com állítása szerint — a hálózati iparág legszélesebb választékát ez a család nyújtja a kisebb irodák integrált hálózati megoldásaihoz. Az új termékbejelentésekkel együtt néhány más fejlesztési eredmény is „bekerült" az OfficeConnect rendszerbe: az OfficeConnect ISDN LAN Modem és 4, 8, illetve 12 portos Fast Ethernet hub. Az új 400-as típus valódi plug-and-play négyportos kapcsoló, amely automatikusan érzékeli az átviteli sebességet. A 280-as típusú kapcsolót a felhasználók „azonnali" teljesítményigényeire méretezték, egyszerűen installálható és használható. Tíz portjából a két 10/100 Mbit/s-os autoérzékelős kivitelű. Ez a két port leginkább nagysebességű szerverkapcsolatokhoz, nagy teljesítményű munkaállomás-összeköttetésekhez és nagysebességű gerinchálózatokhoz nyújt költséghatékony megoldást. Az új OfficeConnect kapcsolók annak az 1996 februárjában bevezetett, kis irodák hálózati funkcióit teljes körűen megoldó OfficeConnect családnak a tagjai, melyekkel máig több mint hétmillió kapcsolatot tartanak fenn világszerte.

Token Ring: Eleve Gigabit/s-ra

Az IBM bejelentette a 100 Mbit/s sebességű Token Ring eszközök megjelentetését. Az első berendezések a jövő év első felében várhatók. Ugyanakkor a Gigabit Token Ring fejlesztése is megkezdődött. A Token Ring igazi erősségei a nagy sebességű hálózatokon válnak nélkülözhetetlenné. A Gigabit sebességű campus gerinchálózatok építése egy sor kihívást jelent. Hogyan használhatók a Gigabit csatornák hatékonyan, amikor még a legnagyobb Ethernet keretméret esetén is kb. 80 ezer frame/s szükséges a csatorna kitöltéséhez? Hogyan tudja az Ethernet a szolgáltatási minőség, illetve szolgáltatási osztályok kialakulásának követelményét biztosítani sok kapcsolóból álló gerinchálózaton? Hogyan építhető fel szövevényes (mesh) hálózat OSI2 szintű kapcsolókból? E problémák nem oldhatók meg egyszerűen, mert gyökereik az Ethernet felépítésében rejlenek. A Token Ring esetén ezek a gondok fel sem merülnek. Itt a 16 Mbit/s 1988-as bevezetésekor a maximális keretméret elérte a kb. 18 Kbájtot. A nagyobb keretméret kulcskérdés a nagysebességű LAN-ok jobb kihasználhatóságához. A keretméret növelése drámaian csökkenti a média sebességének eléréséhez igényelt keretfeldolgozási teljesítményt. Mindössze 7000 frame/s képes feltölteni egy Gigabit csatornát a Token Ring keretméretével. A hálózati forgalom prioritásainak szabályozása is alapvető fontosságú. Ezt már a Token Ring eredeti specifikációja támogatja. Mivel Gigabit kapcsolatokon is megfizethető, 100 Mbit/s portok esetén pedig egyenesen olcsó, az előrelátó hálózattervező a kapcsolók közé kereszt-összeköttetéseket is beépít, amellyel kiküszöbölheti az ugyanazon ponton fellépő hibák hatását. Amíg az Ethernetben tiltott a többszörös aktív utak egyidejű működése egy hálózaton, a Token Ring Source Bridging támogatja a szövevényes hálózatstruktúrák használatát. A Token Ring architektúrájával megelőzte korát, most pedig már elérkezett az ideje a benne rejlő előnyök kihasználásának.

Lantronix: Ethernet eszközbővülés

A legtöbb hálózatban találhatók olyan eszközök, amelyek soros vonalakat igényelnek a monitorozásra és a felügyeleti funkciókra. A Lantronix Micro Serial Server (MSS) elnevezésű, kis helyen elférő, a hálózat elemeihez kapcsolható külső eszközei összekötik ezeket a soros működésű berendezéseket — legyen az terminál, PBX rendszer, vonalkódolvasó vagy Unix szerver — az Ethernet hálózattal. Az MSS-ek a TCP/IP, IPX és LAT protokollokat egyaránt támogatják, megengedve az adatátalakítást bármely hálózati erőforrás-számítógép és az MSS szerverhez kapcsolt berendezés között. Az MSS-ek maximum 115,2 kbit/s-os soros átviteli teljesítményre képesek, és arra, hogy automatikusan alkalmazkodjanak a soros eszköz adatátviteli sebességéhez. Az MSS közvetlenül menedzselhető a szerver fizikai portján vagy egy másik hálózati pontról. Az eszközök támogatják a TCP/IP Digital TSM, DECnet és az SNMP protokollokat. Az MSS1-T típus 10Base-T Ethernet interfésszel RJ45 és RS232 típusú csatlakozókkal van kialakítva. Az MSS1-T2 rendelkezik mind a 10Base-T, mind a 10Base2 Ethernet csatolókkal és RS232 soros kimenettel. A legújabb MSS485-T-hez Ethernet 10Base-T interfész és egy RS485 RJ 45 soros port tartozik. A Lantronix a kapcsolók terén széles termékkínálattal jelentkezett: a sima Ethernet (LMS8, LSB4, LB2) és a kompakt LMS típusú tízportos 10/100 Ethernet kapcsolótól kezdve a nagyobb LW8F eszközökön át a menedzselhető LSW8F-S-ig már bemutatott ilyen termékeket (Új Alaplap, 1998. február). A legújabb az LSW4F-16-MS, amely ez év márciusában jelent meg. Ez egy 20 portos 10/100-as, menedzselhető Ethernet-kapcsoló, ebben négy darab, 10/100-as automatikusan kapcsolatot felvevő és tartó RJ45 port és 16 darab 10 Mbit/s-os port támogatja az SNMP, RMON, Telnet és soros port konfigurációjú menedzsmentet.

NetWare-hez csatolt Oracle8

Az Oracle és a Novell bejelentette, hogy új egyezményük értelmében a Novell NetWare felhasználói megkapják az iparág vezető adatszerverét, az Oracle8-at. Ötfelhasználós Oracle8-at tartalmaz minden NetWare 4.11 és afeletti változat. Az eszközkombináció révén a NetWare-felhasználók hatékony alkalmazáskörnyezetet kapnak, amelyen az üzleti folyamatokat áramvonalasító hálózati számítástecnikai megoldásokat — közöttük Java-alkalmazásokat — fejleszthetnek ki. A termékpárosítás egyetlen dobozban tartalmazza az adatbázis-megoldások fejlesztéséhez szükséges teljes integrált alkalmazáskörnyezetet. Ennek a termékkapcsolási egyezménynek eredményeként olyan hálózati számítástechnikai környezet jön létre, amely csökkenti a rendszer bonyolultságát, a felhasználókat gyorsabban és biztonságosabban engedi hozzáférni a kritikus fontosságú adatokhoz, javítja a hálózati erőforrások felügyeletét, és csökkenti az összköltséget. A termékkapcsolásban a teljes, minden korlátozástól mentes Oracle8 szerepel. Mindkét cég külön-külön beépítette a Javát technológiáiba, így a fejlesztők könnyen készíthetnek hordozható hálózati számítástechnikai alkalmazásokat. Fejlesztők és külső szoftvergyártók által készített mintaalkalmazások is találhatók majd az Oracle8 NetWare-termékcsomagban. A termékkapcsolás a NetWare 4.11-re ez év júniusában lép érvénybe, így az Oracle8-at azonnal tartalmazni fogja a NetWare 5. Hírek szerint a NetWare 5-ben lesz a világ leggyorsabb Java környezete. A Key Labs által végzett független vizsgálatok szerint a NetWare 5 Java Virtual Machine (JVM) több mint kétszer gyorsabb, mint legközelebbi vetélytársa, a Microsoft JVM-je NT Server alatt. E teszteket egy Intel 300 MHz-es Pentium II-es, 128 megabájt RAM-mal felszerelt gépen, 200 kapcsolatot létesítve végezték. Az előzetes vizsgálatok azt jelzik, hogy a Novell JVM-je többszörösen gyorsabb, mint a vezető Unix rendszerek Java-implementációi hasonló árú hardveren.

Kovács Attila

A cég eltűnt, az eredmény megmaradt

Novonyx-átváltozások

A számítógépes hálózati piac szereplőinek termékei között előfordul, hogy egymást kiegészítve tudnak jobb szolgáltatást nyújtani. Ilyenkor a termékfejlesztők piaci érdekei is abba az irányba mutatnak, hogy inkább az együttműködés kerüljön előtérbe. Ebből a felismerésből fakadva jött létre 1997-ben a Novonyx cég is. Alapítói a Netscape Communication és a Novell, melyek külön-külön is jelentős hálózati szoftverfejlesztők.

S hogy hol találkoznak ezek a technológiák? Az intranet rendszerekben. Ezekben egyesül a zárt hálózatok hagyományos technológiája számos olyan elemmel, melyek először a nemzetközi nagy hálón, az Interneten jelentek meg. Az intranet hálózatok belső adatforgalma egyre gyakrabban az Interneten megszokott, először azon bevezetett módokon, formátumokban zajlik. Ezzel párhuzamosan az intranet rendszerek kialakításakor egyre inkább az a tendencia válik uralkodóvá, hogy a cégek belső hálózatairól közvetlen kijárás legyen az Internet felé is.

Ezen a találkozási ponton határozta el a két alapító az új cég létrehozását. A Novonyx megalakítására az első tervek és szándéknyilatkozatok közel egy éve láttak napvilágot. Azután az önálló cég megkezdte működését. S itt a vége... Mármint az önálló Novonyx cégnek, mivel részvényeit a Novell megvásárolta a Netscape-től, és ezzel az új cég visszaolvadt a Novellbe.

De nincs végük a közösen tervezett termékeknek, a fejlesztési koncepcióknak. A közös irányvonal elkészült termékei pedig mindkét hálózatos cég, a Novell és a Netscape üzleti polcain egyaránt megtalálhatók. Annak idején az új cégnek első üzleti céljai között szerepelt a Netscape Internet szerverekre kifejlesztett SuiteSpot programcsalád adaptálása, integrálása, majd elterjesztése a Novell IntraNetWare platformjain alapuló belső hálózatokon. Ezzel akartak a Novell NetWare hálózati operációs rendszert alkalmazó felhasználónak világszínvonalú megoldást nyújtani az Internet-szolgáltatások eléréséhez a hálózatos munkaállomásokról. Ennek a közös fejlesztésnek, forgalmazásnak első lépéseként az IntraNetWare-re készült SuiteSpot alkalmazások forgalmazása volt a cél.

Ezekben az új programokban megvalósítják a Novell címtárrendszerének (Novell Directory Services, NDS) a teljes integrálását, a várakozások és tervek szerint új generációt jelentve az intranetalapú rendszert használó programokban. Ugyanakkor az új SuiteSpot programok támogatni fogják az SMTP, POP3 és IMAP4 Internet-szabványokat, valamint az LDAP-t is.

A közös fejlesztés első „szállítmányai" 1998 januárjában jelentek meg. Ezek között olyan programok szerepeltek, mint a Netscape Enterprise Server 3.0, a Netscape FastTrack Server IntraNetWare-verziói, valamint a Netscape Messaging Server for NetWare.

A teljes Netscape SuiteSpot intranetcsomagot ez év végéig tervezik megjelentetni.

Ezzel párhuzamosan kezdődött meg az együttműködés az Oracle-lel, amelynek célja egyértelműen a hálózatos adatbáziskezelés lehetőségeinek beépítése a nagyvállalati programcsomagokba. Ennek eredményeként az Enterprise Server for NetWare program új, Enterprise Pro Servernek nevezett változatában natív adatbázis-integrációt valósítottak meg. Egyben biztosítva — ahogy a teljes tervezett programcsomagban — a kapcsolatot az NDS-szel. Szintén natív driverek készülnek a Sybase és Btrieve rendszerekhez is, de bármely ODBC adatbázis elérhető JavaScripten keresztül. Ezzel a NetWarep latformokon és weben futó alkalmazások számára egyaránt lehetőség nyílik az adatbázisok elérésére a Netscape Enterprise Webszervereken keresztül. Az Enterprise Pro program NetWare platformról a már említett JavaScript-támogatással biztosítja a webalapú adatbázis-elérést.

A SuiteSpot csomag további tagjai közül az előzetes tájékoztatások szerint 1998 közepére várható a Calendar, a Collabra és a Proxy Server, illetve az év végére a Certificate és Compass Server programok megjelenése. Ezekkel a termékekkel tovább bővülnek a Novell már meglévő Internet/intranet megoldásai (BorderManager, GroupWise). A teljes rendszer támogatást ad külső fejlesztők számára, hogy alkalmazásaikkal kapcsolódjanak a már futó rendszerelemekhez. Ezek fejlesztésére különböző lehetőségek vannak, de a rendszer beépített támogatást tartalmaz a weben leggyakrabban használt nyelvekhez (Java, JavaScript, Perl), és a Novell által fejlesztett NetBasic-hez is.

Simay Endre István

Novell: Netscape FastTrack Server for NetWare Competitive Chart - Netscape

File Edit View Go Communicator Help

Comparative Matrix

Company Name	Internet Factory	Microsoft	Novonyx	O'Reilly	Novell	Novonyx
Product	Commerce Builder 2.1	IIS 3.0	Enterprise Server for NetWare	WebSite Professional	Web Server 3.0	FastTrack Server
Prices	$395	Included with NT	$1,295	$499	$295	$295
Features						
GUI Installation	Y	Y	Y	Y	N	Y
Virtual Servers	Y	Y	Y	Y	N	N
Server Side Includes (SSI)	Y	Y	Y	Y	Y	Y
Search Engine	N	Y	Y	Y	Y	N
Virtual Directories	Y	Y	Y	N	Y	Y
HTML Authoring tool	N	Y	Y	Y	Y	Y
Image mapping utility	N	Y	Y	Y	Y	Y
Supports NetWare Platform	N	N	Y	N	Y	Y
Management						
Remote administration	Y	Y	Y	Y	Y	Y
SNMP support	N	Y	Y	Y	Y	N
Integration With NDS	N	N	Y	N	Y	Y
Browser administration	Y	Y	Y	N	Y	Y

Hálózati vetélkedő 2.: Skálázhatóság

Technológiák révén

Napjainkban a vállalatok túlnyomó részénél nő az informatika szerepe, hiszen egyre több funkcionalitást valósítanak meg számítógépekkel. Az egyre nagyobb adatbázisok kezelését, az újabb alkalmazások által igényelt megnövekedett teljesítményigényt természetesen a rendszerek lelkét jelentő szervereknek is követniük kell. A meglévő beruházások védelmében a cégek elvárják számítógépeiktől, hogy változatlan konfigurációval vagy kis bővítéssel kövessék a vállalati igényeket. Az, hogy a felhasználószám növekedése vagy egy-egy újabb alkalmazás, újabb funkció bevezetése együttjárjon új szerver üzembe helyezésével, a legtöbb cégnél elfogadhatatlan.

A Novell NetWare-verziók tervezésénél a skálázhatóság az egyik igen hangsúlyos szempont. Az, hogy egy operációs rendszer éppoly hatékonyan és gyorsan szolgáljon ki öt felhasználót, mint ezret, csak akkor érhető el, ha már a rendszer tervezésénél és megvalósításánál a hálózati szolgáltatások minél hatékonyabb megoldására lehet összpontosítani. A Novell fejlesztései az első NetWare-verziók megjelenésétől kezdve ennek figyelembevételével történtek, ezért a háttérben dolgozó NetWare szerverek a legtöbb cégnél folyamatosan és megbízhatóan nyújtják a tőlük eredetileg elvárt szolgáltatásokat.

Mik is ezek a szolgáltatások? Nyomtatómegosztás; fájlok központi tárolása és elérése akár adatok, adatbázisok esetén, akár programoknak a szerverről való indítása során; adatbázisszerver funkció; internetes szolgáltatások (web, ftp); címtárszolgáltatás stb.

Menedzselés is

Ez a megközelítés és a sokéves fejlesztési tapasztalat a hálózati szerverek piacán talán a legfontosabb tényező abban, hogy egy IntraNetWare szerver gond nélkül képes kiszolgálni több ezer felhasználót is. Magyarországon is vannak erre példák, és az egyik ilyen helyen a több mint ezer felhasználó fele lokális diszk nélküli PC-ken dolgozik. Azaz a felhasználók kiszolgálása nem azt jelenti, hogy azok naponta elolvasnak vagy módosítanak két-három állományt a szerveren, hanem onnan indítják gépeiket, töltik be az operációs rendszert, és természetesen minden alkalmazásuk és adatuk is ezen a központi gépen található.

Természetesen a skálázhatóság nemcsak a szoftveren múlik. Ahhoz, hogy az egyre nagyobb igényeket ki lehessen elégíteni, a hardverelemeknek is lépést kell tartaniuk a fokozott elvárásokkal. A hardvergyártók egyre újabb technológiákat fejlesztenek ki, és a gépekbe egyre több memória, egyre több és nagyobb teljesítményű processzor kerül. A hálózatok sebessége is rohamosan nő, a 10 Mbites és 100 Mbites Ethernet hálózat mellett már megjelent a Gigabit Ethernet, és az ATM is egyre jobban tért hódít.

Mit tud tenni ezen a területen az operációs rendszer gyártója? Két dolgot. Egyrészt támogatja az új eszközöket, új szabványokat. Természetesen a NetWare is ezt teszi, például a többprocesszoros szervereket egészen 32 processzorig támogatja. Ugyanez érvényes az ATM-re és a Gigabit Ethernetre is. Ez a dolog egyik oldala.

Nem elég persze egyszerűen csak támogatni a új hardverlehetőségeket a megfelelő driver megalkotásával, hanem ki is kell használni az új eszközök által nyújtott előnyöket, elérve a lehető legnagyobb teljesítménynövekedést. Erre jó példa a Gigabit Ethernet. Ez a technológia még csak kísérleti stádiumban van, de az IntraNetWare már támogatja, és nemcsak támogatja, hanem képes is kihasználni a kártya által nyújtott sávszélességet. Végeztek egy tesztet ezzel kapcsolatban (Internet Week, 1997. október 6-i szám), és a vizsgált rendszerek közül csak az IntraNetWare és a Sun Solaris tudta „kihajtani" a kérdéses kártyát.

A skálázhatósággal kapcsolatban még egy dolgot kell megemlíteni, a menedzselhetőséget. A legtöbb helyen a processzor órajelének értékei, a memóriaméret, a multiprocesszoros támogatás jelenti a skálázhatóságot. Nem elég azonban kiválasztani az erős szervergépet, amely lehetővé teszi ezer felhasználó kezelését, hanem azt az ezer felhasználót be is kell kapcsolni, és menedzselni kell őket. Erre a Novell címtár, az NDS nyújt megoldást. Az NDS révén egy strukturált, akár a vállalat szervezeti felépítésére ráilleszthető adatbázisban lehet az összes felhasználót menedzselni.

A jövő útja

Az NDS tette azt is lehetővé, hogy a Novell pár hét alatt fel tudta építeni a hannoveri CeBIT után a világ második legnagyobb informatikai rendezvényének, a Las Vegas-i Comdexnek a kiszolgáló hálózatát. Hat darab szerver kezelte a Comdex 220 000 (!) látogatóját, valamennyiüket egyetlen központi címtárban szerepeltetve, az NDS pedig lehetővé tette az összes látogatónak a rendszerhez való hozzáférését. Erre az NDS-re épült rá a GroupWise, a Novell csoportmunka-támogató szoftvere, amelynek segítségével a kiállítás látogatói levelezni tudtak. A GroupWise minden látogatónak naponta továbbította az előadásokra szóló meghívókat, a programokat és az egyéb közérdekű információkat. És természetesen ezen a rendszeren keresztül kaptak visszajelzést a rendezvény szervezői a látogatóktól.

Végezetül nézzük, hogyan látja a skálázhatóság jövőjét a Novell. A szerverektől elvárt egyre nagyobb teljesítményt a clustering technológia révén lehet majd nagyon egyszerűen kielégíteni. A Novellnek már most is vannak ilyen megoldásai (SFTIII, Standby Server), amelyekkel hibatűrő rendszereket tud kiépíteni. Orion néven 1998 második felében várható az az új clusterrendszer, amely a hibatűrés mellett teljesíménynövekedéssel is jár majd. Maximum 16, akár különböző gyártóktól származó szervert is össze tud fogni. A gépek tejesítménye összeadódik, és az egész rendszer egyetlen hatalmas kapacitású számítógépnek látszik majd.

Hargitai Zsolt

Hálózati vetélkedő 2.: Skálázhatóság

Lefelé és felfelé

Egy kiszolgáló operációs rendszert akkor nevezünk jól skálázhatónak, ha az mindössze néhány egyidejű felhasználót kezelő, alacsony számításigényű, vagy éppen kevés adatmozgatással járó feladatokra ugyanúgy alkalmas, mint a több ezer konkurens kérést hatékonyan végrehajtó, szuperszerveri képességeket igénylő működésre. A Windows NT technológia és ezen belül a Windows NT Server felépítése és szolgáltatása az informatikai ipar egyik legjobban skálázható, ugyanakkor rendkívül költséghatékony megoldása.

Habár az operációs rendszerek szállítói általában a nagy számok bűvöletében élnek, fontos megjegyezni, hogy egy platformot nemcsak felfelé, hanem lefelé is skálázni kell.

— A Windows NT Server az egyprocesszoros kiszolgálóktól a nyolc-tizenkét CPU-t tartalmazó középszámítógépekig több ezer különböző géptípuson futtatható.

— Az azonos technológiára épülő, de a gép előtt ülő felhasználót figyelembe véve eltérően hangolt Windows NT Workstationt használják a NetPC-ken, a hagyományos és hordozható személyi számítógépeken és a professzionális, akár többprocesszoros munkaállomásokon.

— A Windows CE, a „PC alatti" hardverkategória (kézi, telefonba és autórádióba integrált számítógépek, webtelevíziók, Windows terminálok) operációs rendszere szintén a Windows NT technológiából született.

— De a lánc még itt sem szakad meg: a Windows NT kernel az alapja az ipari folyamatirányító eszközök új generációjának is.

A rendszer felépítése

A rendszerplatform fent említett változatait integrálja az egységes Win32 programozási felület, a COM komponenstechnológia, a hálózati és B/K szolgáltatások halmaza, a közös felügyeleti modell, és így tovább.

Ennek a széles skálázhatóságnak az alapjai az operációs rendszer felépítésében rejlenek. A Windows NT Server közbeavatkozó többfeladatos (preemptive multitasking) operációs rendszer. Ez azt jelenti, hogy a rendszer- és alkalmazásfolyamatok (process) egy egy meghatározott időszeletet kapnak. Ha ez lejár, a kernel elveszi az éppen futó folyamattól a vezérlést, és a következőnek adja. A folyamatok így látszólag egyidejűleg futnak. Ha pedig a számítógép több processzorral rendelkezik, akkor valóban párhuzamosítható a végrehajtás. Mi több, a Windows NT Server többszálú (multithreaded) operációs rendszer, amely egyes rendszerkomponenseket és alkalmazásokat a folyamatoknál finomabb ún. szálakra bont. Ezzel a megoldással egyetlen alkalmazás különböző részeinek párhuzamos végrehajtására is lehetőség nyílik.

Eleve több processzorra

A jól skálázható operációs rendszerek egyik titka tehát a különböző számú processzort tartalmazó hardverplatformok egyformán hatékony kezelése. A cikksorozat előző részében már említettük az adatbáziskezelő platformok teljesítménytesztjeit hitelesítő Transaction Processing Councilt. A szervezet http://www.tpc.org címen elérhető webhelyén egy- és kilencvenhat processzoros gépet egyaránt találhatunk.

Érdemes megfigyelni, hogy a lista négyötödét kitevő egy, kettő, négy, hat és nyolc CPU-t tartalmazó rendszerek közel 50%-a Windows NT Servert futtat. Az egyes platformok legjobb Windows NT-alapú eredményeit elemezve látható, hogy az operációs rendszer mint alkalmazásplatform gyakorlatilag lineáris skálázhatóságot biztosít, amelynek mértéke átlagosan 84 százalék.

Rendkívül lényeges, hogy a több processzorból felépített számítógépek teljesítménytöbbletét nemcsak a speciális módon megírt alkalmazások használják ki. Ellentétben az eredetileg egy CPU-ra készített és utólag átalakított rendszerplatformokkal, a Windows NT Server eleve szimmetrikus többprocesszoros működésre (symmetric multiprocessing, SMP) tervezett platform, amely bármely alkalmazást képes más programokkal és a rendszerrel párhuzamosan futtatni, és egyenlően terheli az egyes CPU-kat. A processzorkezelésen túl aszinkron B/K rendszere, hatékony helyi és távolsági hálózatkezelése is hozzájárul a skálázhatósághoz. Az 1997. május 20-án megrendezett Scalability Day nevű rendezvényen a Microsoft többek között az alábbiakat demonstrálta:

— Egy terabájtnyi információt (Földünk valamennyi 25 000 főnél népesebb településének műholdképét) tároló és elérhetővé tevő Terra-Servert, amely egyetlen számítógépen, Windows NT Servert és SQL Servert használva futott.

— Napi százmillió „találat" (hit) kiszolgálására képes webkiszolgálót, szintén egyetlen, Windows NT Servert és Internet Information Servert futtató gépen.

— Napi 1,8 millió üzenet továbbítására, 50 000 felhasználó kiszolgálására és 50 gigabájtnyi üzenettár kezelésére szolgáló Exchange Servert, amely egyetlen gépen, Windows NT Server operációs rendszeren futott.

Fürtös megoldás

A kiszolgáló újabb és újabb központi egységekkel való bővítése csak az egyik lehetőség a teljesítmény növelésére. Ha több számítógépet ún. fürtbe (cluster) kapcsolunk, feldolgozóképességük összeadódhat.

A Scalability Day során a Microsoft bemutatott egy húsz, egyenként négy processzort tartalmazó kiszolgálóból álló fürtöt. A Windows NT Servert, Transaction Servert és SQL Servert futtató számítógép-együttes egymilliárd üzleti tranzakciót hajtott végre egyetlen nap alatt. A fenti teljesítményteszt és a Scalability Day egyéb bemutatói a http://www.microsoft.com/backoffice/scalability/demo.htm címen megtalálhatók.

Kőnig Tibor

Hálózati vetélkedő 2.: Skálázhatóság

Rutinból...

„Skálázhatóság” — a számítástechnikai ipar egyik legutóbbi, legdivatosabb kulcsszava. Talán nem is tudjuk, mit nyújt nekünk, de azt már hallottuk, hogy szükségünk van rá. Egyszer csak minden gyártó azt hiszi, hogy nagyon fontos dologról van szó, és tolonganak, hogy ők bizony biztosítani tudják a felhasználóknak a skálázhatóságot.

Mi az IBM-nél nem igazán tudjuk, hogy mi most ez a nagy felhajtás a skálázhatóság körül. A skálázhatóság egyáltalán nem új az IBM felhasználói számára, hiszen az IBM felhasználóival együtt már akkor „skálázhatóságban dolgozott”, amikor a skálázhatóság még nem volt divat.

Fogalomtisztázás

Mi is valójában a „skálázhatóság”?

— Lehetővé teszi, hogy alkalmazásokat futtassunk olyan számítógéprendszeren, amely megfelelő méretű vállalatunk számára, akármekkora is a vállalat, és lehetővé teszi, hogy rendszerünket problémamentesen nagyobbra vagy kisebbre méretezzük jelenlegi eszközeink megtartásával.

— A skálázhatóság fogalmához tartozik az is, hogy a fenti jellemzővel bíró rendszerek és szoftverek folyamatosan, megszakítás nélkül, zavartalanul működnek, akár 10, akár 10 millió felhasználóról van szó.

A nagyvállalati számítástechnika valós világában a skálázhatóság sokkal több, mint összekötni több számítógépet. Megvalósítani a valós világban értett skálázhatóságot magában foglalja az üzleti-kritikus rendszerek tízezreit, amelyek utasításokat hajtanak végre kevesebb mint két másodperces válaszadással egyidejűleg több ezer felhasználónak kiszolgálónként. Mindezt teljesítménycsökkenés nélkül. Megbízhatóan, megfelelő integritással és biztonsággal.

Az IBM felhasználói különböző platformokat használnak az üzleti életükben: IBM szoftver IBM hardveren, nem IBM IBM-en, IBM nem IBM-en. Felhasználóink már régóta használnak különböző skálázhatósági megoldásokat, amelyek IBM operációs rendszerekbe és szerverekbe vannak integrálva. Ezek az IBM rendszerek támogatnak egyidejűleg 10, vagy akár 150 000 felhasználót. Ezek a rendszerek biztosítják a megbízhatóságot, a rendelkezésre állást és a biztonságot azoknak a felhasználóknak, akiknél követelmény, hogy nagyobb rendszerek, nagyobb hálózatok, intranet- és Internet-alkalmazások irányába tudjanak mozdulni. Ez az, amit mi skálázhatóságnak hívunk.

De nézzük, hogyan is néz ki ez „kicsiben”, az OS/2 Warp Server esetében. A skálázhatóság számunkra igen nagy területet fed le. Skálázhatóság alatt a vállalat egészére kiterjedő rendszert értünk, ami lehet teljesen heterogén hálózatokból és platformokból felépített rendszer is. Az OS/2 Warp Server tisztán illeszkedik ebbe a képbe, kapcsolatot tartva nagyobb rendszerekkel, AS/400, RS/6000 Unix rendszerekkel vagy akár S/390 mainframe-mel. De önmagában is képes önálló rendszert képezni.

Tükrözéssel

A fürtözés és a multiprocesszoros környezet két olyan technológia, amelyet mostanában vezettek be Intel platformokra, növelve ezzel a kiszolgálók teljesítményét, a skálázhatóságot és a rendelkezésre állást.

Az IBM a Vinca cég (http://www .vinca.com) StandbyServer for OS/2 termékét ajánlja fürtözési megoldásként. Ez tükrözési rendszer, amellyel egy tartalék kiszolgálót biztosíthatunk az elsődleges mellé.

Ha elsődleges kiszolgálónk megállna, akkor a másik gép automatikusan át tudja venni az elsődleges kiszolgáló teljes szerepét. A kliensek kapcsolata az egyik kiszolgálóról a másikra való átkapcsolás alatt megmarad, és csak egy pillanatnyi késleltetést vesznek észre. A munka nem szakad meg, nem történik adatvesztés.

Kiszolgálóleállások három fő ok miatt következhetnek be: a hardver, az operációs rendszer vagy az alkalmazás problémái miatt. A Vinca StandbyServer termékét integrálva az IBM Netfinity termékével (mely beépített része a Warp Servernek) a három fő probléma mindegyikét figyelhetjük, és utasíthatjuk, hogy különböző eseményeket hogyan kezeljen, mikor szükséges a „háttérkiszolgáló” indítása.

Ezek a funkciók teljesen automatikusan történnek, manuális beavatkozásra nincs szükség. Kérhetjük a rendszert arra is, hogy a kiszolgálón felmerülő problémákról értesítse a rendszergazdát (e-mailben, csipogóra stb.).

A valós idejű tükrözés tranzakciókon alapszik, így nem szenved adatkésleltetést, mint például egy normál fájlmásolás. A tükrözés egy dedikált hálózati kapcsolaton történik, tehát nem okoz felesleges forgalmat a hálózaton. Előnye még, hogy a kiszolgálóknak nem kell ugyanabban a szobában lenniük, illetve nem szükséges, hogy a kiszolgálók bármilyen szempontból is azonos hardverfelépítésű gépek legyenek.

Szerepcsere

A Vinca-féle StandbyServer megoldásnak van egy továbbfejlesztett változata is, melyet nagy adatmennyiséggel rendelkező vállalatoknak ajánlunk. Ez a Netfinity Cluster Pack, amely nem csupán a Vinca StandbyServer szoftverét tartalmazza, hanem kétirányú kapcsolatot is létesít két kiszolgáló között. Következésképpen ezzel a rendszerrel mindkét számítógép dolgozhat kiszolgálóként, és mindkettő kész arra, hogy a másik szerepét átvegye, ha arra szükség van.

Ez a megoldás az IBM SCSI 2 adapterén alapszik, mely lehetővé teszi, hogy egy számítógép hozzáférjen a másik számítógép RAID környezetében tárolt adatokhoz.

Az IBM régóta vezető szerepet játszik az SMP technológiában. Az IBM volt az első cég, amely 1993-ban az OS/2 2.11-es verziójával SMP-megoldást szállított tisztán PC-s környezetbe. Az OS/2 Warp Server SMP 2-4 processzorra van optimalizálva, hiszen ezek napjaink domináns SMP rendszerei, de architekturálisan 64 processzort tud kezelni.

Pál Ferenc

Hálózati vetélkedő 2.: Skálázhatóság

Teljes biztonsággal

SCO UnixWare

Egy számítógép vagy szoftver a vásárlás pillanatában adott feladatok elvégzésére alkalmas, kérdés azonban, hogy idővel, ahogy igényeink vagy a környezet által támasztott igények nőnek, a gép (vagy a rendszer) mennyire tud ezekkel együtt nőni.

Ha egy termék jól skálázható, akkor évekig tovább lehet bővítgetni az igényeknek megfelelően, ha nem skálázható, akkor le kell cserélni másik termékre. Egy nagy logisztikai rendszernél, amelyet huszan, harmincan, százan használnak, egy ilyen váltás hatalmas ráfordításokat igényelhet (beüzemelés, betanítás, kezelési és működési hibák elhárítása...).

De a skálázhatóságot sem adják ingyen! A skálázható rendszerek kulcskérdései az alábbiak:

- » **Hány felhasználót tud a rendszer maximálisan kezelni?**
- » **Hány CPU-t tud maximálisan kezelni? (Hardver is, szoftver is.)**
- » **Többszálú-e az operációs rendszer?**
- » **Mennyire bővíthető a memória? (Hardver is, szoftver is.)**
- » **Az alkalmazás nem dugul-e be, ha sokkal többen használják? (Ekkor már a legjobb hardver sem segít.)**
- » **Mennyire képes a rendszer az új eszközöket kezelni? (Kártyák, hálózatok, perifériák, protokollok, szabványok...)**
- » **Át lehet-e vinni más, erősebb hardverre az alkalmazást, egyszerűen, átírás nélkül?**
- » **Az adott hardveren lehet-e mást is futtatni, párhuzamosan az adott alkalmazással? (Mennyire nyitott az architektúra?)**
- » **Mennyire lehet a rendszer futási biztonságát növelni? (Diszktükrözés, rendszertükrözés, fürtözés stb.)**
- » **Mennyire lehet a rendszer hozzáférési biztonságát növelni?**

A skálázhatóság a megbízhatóság mellett egyre fontosabb szemponttá vált az üzleti kiszolgálók és az azt működtető operációs rendszerek beszerzésekor. A felhasználó általában jelenlegi, illetve tervezett közeljövőbeli igényei alapján dönt a vásárlásról. Azt maga sem tudja, hogy 1 vagy 2 év múlva a rendszerrel szemben milyen követelmények lesznek. (Például az erőforrásokat illetően.)

Jelentős erőforrástöbbletre lehet szükség, ha a felhasználók száma növekszik, ha módosítanak a meglévő alkalmazáson, esetleg az adott adatbáziskezelőnak újabb változatát kell használni, vagy további alkalmazásokat kell telepíteni a rendszerre.

Ha a felhasználó kinőtte a rendszer kapacitását, és az tovább nem bővíthető, bizony rengeteg probléma adódhat. Először is, ha a rendszer nem régi, akkor meg sem térült a belefektetett pénz, máris le kell selejtezni. Kérdés ilyenkor, hogy lehet-e másra használni. Ez mindenesetre az Intel-alapú gépek mellett szól, hiszen egy ilyen szervert később oda lehet adni a titkárnőnek is, mert kedvenc szövegszerkesztőjének 2002-es változata is minden bizonnyal használható rajta, de ugyanez mondjuk egy IBM AS/400-assal kicsit már problémás.

Az alkalmazásról tényleg semmit sem lehet mondani azon kívül, hogy a fenti szempontok alapján alaposan végig kell nézni az egyes tulajdonságokat.

Utána jön az operációs rendszer. A fenti követelményeknek a kapható operációs rendszerek jelentős része valamilyen mértékben eleget tesz. Kiemelkednek azonban a Unix rendszerek, amelyekre mindig is jellemző volt a nagyfokú skálázhatóság. Ma a Unixok alkalmasak akár egy-két, akár többezer felhasználó kiszolgálására. Itt a választék a nem professzionális célokra szánt, gyakorlatilag ingyenes Linux rendszertől (PC hardver) a szintén Intel alapokon futó professzionális Unixware-en vagy SPARC (RISC) processzoron futó Solarison keresztül egészen a 4000 processzor párhuzamos kezelésére képes Tandem Unixig terjed. Már a Linux is képes egy komolyabb teljesítményű gép kezelésére, de például a professzionális célokra szánt SCO kapható egyfelhasználós változatban is, és elindul akár egy 486-os PC-n, 16 MB RAM-mal, 850 MB-os diszkkel, ugyanakkor teljes kiépítésben a Unixware7 olyan rendszerré fejleszthető, amely 1000-2000 felhasználót is kiszolgálhat. A kezelhető processzorok száma: 32, a maximális RAM-méret 64 GB, a maximális diszkméret 76,5 TB.

Ez az operációs rendszer egyebek között pl. szoftverből tudja különböző RAID szinteken védeni a diszkeken lévő adatok biztonságát. Emellett fürtözés (clustering) segítségével több gépet össze tud kapcsolni úgy, hogy azok egymás erőforrásait használják. Párhuzamosan futtatható 2, 4 vagy akár 6 gép is, és azok képesek egymás feladatait átvenni, ha az egyik valami miatt meghibásodna. A fürtözés másik feladata a terhelés megosztása. Ekkor a fürtözést alkotó gépek egyetlen közös memóriaképet láthatnak (Single System Image = SSI) és az alkalmazások transzparensen férhetnek hozzá bármelyik gép memóriájához és perifériájához.

A fenti tényezők már hardveres terepre vezetnek át. Évek óta látszik, hogy a multiprocesszoros rendszerek egyre népszerűbbek. Ezek óriási előnye, hogy néhány kártya bedugásával a szerver teljesítményét félóra alatt többszörösére növelhetjük, szemben a szervergép esetleg több napig vagy egy hétig is tartó cseréjével, ami egy informatikai vezető rémálma. Azt azonban tudomásul kell venni, hogy egy többprocesszoros rendszer egy-két processzorral fajlagosan jóval drágább, mint az ugyanezt tudó, de tovább már nem bővíthető rendszer. Emellett a processzorok hatékonysága is nyilván kisebb, ha egy 8 processzorra tervezett gépben 2 processzor „lötyög". Itt a választék széles, mert 2, 4, 8, 12, 64 processzoros gépek is elérhetők Intel, PowerPC, Sparc, MIPS processzorokkal. Nemzetközi tapasztalatok szerint a nagy cégek ilyen teljesítménytartományokban előnybe részesítik a Unix rendszereket, mert egyrészt az ár/teljesítmény viszony kedvező, másrészt ezen a területen a Unix rendszerek már bizonyították, hogy az elengedhetetlenül szükséges nagyfokú megbízhatóságot garantálni tudják, márpedig egy olyan rendszernél, amely létkérdés a cég számára, ez (kellene hogy legyen) a legfontosabb tényező.

Toporczy István

Memorizálunk... — IV.

Az SDRAM házatáján

Az Intel LX chipkészlete az első Pentium II chipkészlet a gyors szinkron DRAM-hoz (SDRAM). Ezenkívül fogadja az EDO RAM-ot is, de a régi Fast Page Mode RAM-ot (FPM) nem. Mivel az SDRAM kizárólag DIMM kivitelben (Dual Inline Memory Module; 168 lábú, mindkét oldalán kontaktussal ellátott formában) készül, a gyártónak megfelelő foglalatot kell az alaplapra építenie. Alábbi írásunk áttekintést próbál adni a technikai jellegű ismeretekről.

Az SDRAM kifejezés a szinkron DRAM-ból származik. Kifejlesztésének megértéséhez értenünk kell azt az alapproblémát, hogy a „hagyományos" RAM-ok és az EDO RAM-ok jóval lassabbak, mint a CPU, ezért a CPU sokáig várakozik, mire egy utasítás vagy adat a memóriából végre megérkezik.

Miközben a CPU fejlődése töretlennek mondható — azaz a Moore-szabály szerint az Intel általában 18 hónaponként képes megduplázni az adott chipen elhelyezett tranzisztorok számát —, addig a RAM és különösen a PC buszrendszerének fejlődése lemaradt. A RAM-fejlesztések döntő többsége az elmúlt időkben a RAM-chipekkel függött össze, ebben a jövőben változás várható, és az adatátviteli szélesség is meghatározó lesz.

Egy „hagyományos" FPM RAM minden harmadik órajelnél képes adatot szolgáltatni, egy EDO RAM minden másodiknál, azaz az elért sebességnövekedés kb. 30%-os. Az SDRAM minden felfutó órajelnél szolgáltat adatot, így a „hagyományos" RAM-nál kb. 3-szor gyorsabb. (Csak zárójelben jegyezzük meg, hogy kifejlesztés alatt van a DDR rendszer, amely az órajel felfutó és lefutó élénél egyaránt képes adatszolgáltatásra, így a ma használatos SDRAM-nál kb. 2-szer gyorsabb.)

Bekapcsoltunk...

A PC bekapcsolása után a BIOS egyik első feladata a RAM üzembehelyezése. Ehhez fel kell ismernie, hogy melyik foglalatba milyen típusú RAM van helyezve, és a chipkészletet ennek megfelelően kell „felprogramoznia". Ezután a BIOS megállapítja a RAM nagyságát és szervezettségét. Ezt egy trükkökkel alaposan megtűzdelt algoritmus révén találja ki. Például a chipkészletet különbözően felprogramozza, egy algoritmus szerinti mintát ír a RAM-ba, majd visszaolvassa. Ez az eljárás hosszú ideig is eltarthat. Ha sok RAM van a rendszerben, másodpercek telhetnek el, mire a BIOS végre felismeri a RAM típusát, és a RAM-teszt elkezdődhet.

Az SDRAM-nál már nem elég, ha a BIOS „találgat". Ahhoz, hogy a RAM megfelelően működjön, néhány vezérlőjelet is be kell állítani. Ez attól is függ, hogy milyen az alkalmazott RAM-chipek szervezése. Egy 64 bites modul például felépülhet 8 db x8 szervezettségű chipből, de állhat 16 db x4 szervezettségűből is. Ezt a különbséget azonnal látjuk, ha szemünk pillantását a modulra vetjük. A BIOS-nak azonban nincs szeme, így esélye sem, hogy a RAM típusát ilyen egyszerű módon megállapítsa.

Ravaszkodások

A gyártók szerelhettek volna kamerát is a PC belsejébe, amely betöltés közben a modulra ránéz, ehelyett azonban az Intel fejlesztői a Serial Presence Detect (SPD) eljárást találták ki. Minden modern DIMM-en (mindegy, hogy SDRAM vagy EDO) a RAM mellett található egy kis EEPROM chip. Ezt a chipet az I2C buszon keresztül vezérlik. (Most kihagyjuk az I2C busz tárgyalását.) Az EEPROM chipbe írják a gyártók a DIMM modul jellemzőit: EDO vagy SDRAM, kapacitás, a modul belső szervezettsége, időzítési paraméterek stb. A BIOS az I2C buszon keresztül ki tudja olvasni a modul jellemzőit, és ennek megfelelően a vezérlő chipkészletet fel tudja programozni. Így az időigényes RAM-felismerő eljárás kiesik.

Mivel az EEPROM chipben a szükséges időzítési paraméterek is benne vannak, az alaplap gyártója elvileg nem kell, hogy a felhasználót a gyötrelmes BIOS-beállításokkal kínozza, elég, ha egy „AUTO" felismerést felkínál. Nos, ennyit az elvről. Mint oly sokszor máskor is, a gyakorlatban a dolog kicsit más...

Az első SDRAM DIMM-ek például nem tartalmaztak EEPROM-ot, sőt ma is előfordul, hogy költségkímélés miatt a gyártó nem építi fel az EEPROM-ot a modulon. Az ilyen modulokat az Intel LX specifikációja nem támogatja. Hogy ezek néhány alaplapon mégis működnek, az más lapra tartozik. Az EEPROM nélküli modulokat ránézéssel könnyű felismerni. Az is előfordul azonban, hogy jóllehet a DIMM tartalmazza az EEPROM-ot, de az hibásan van felprogramozva, vagy nem minden adatot tartalmaz. Ez azzal függ össze, hogy idők folyamán az SDRAM specifikációja, illetve az SPD adatstruktúrája megváltozott. Csak 1996 decemberében bocsátotta ki a JEDEC-bizottság az érvényes szabványt és ezzel az SPD adatstruktúráját.

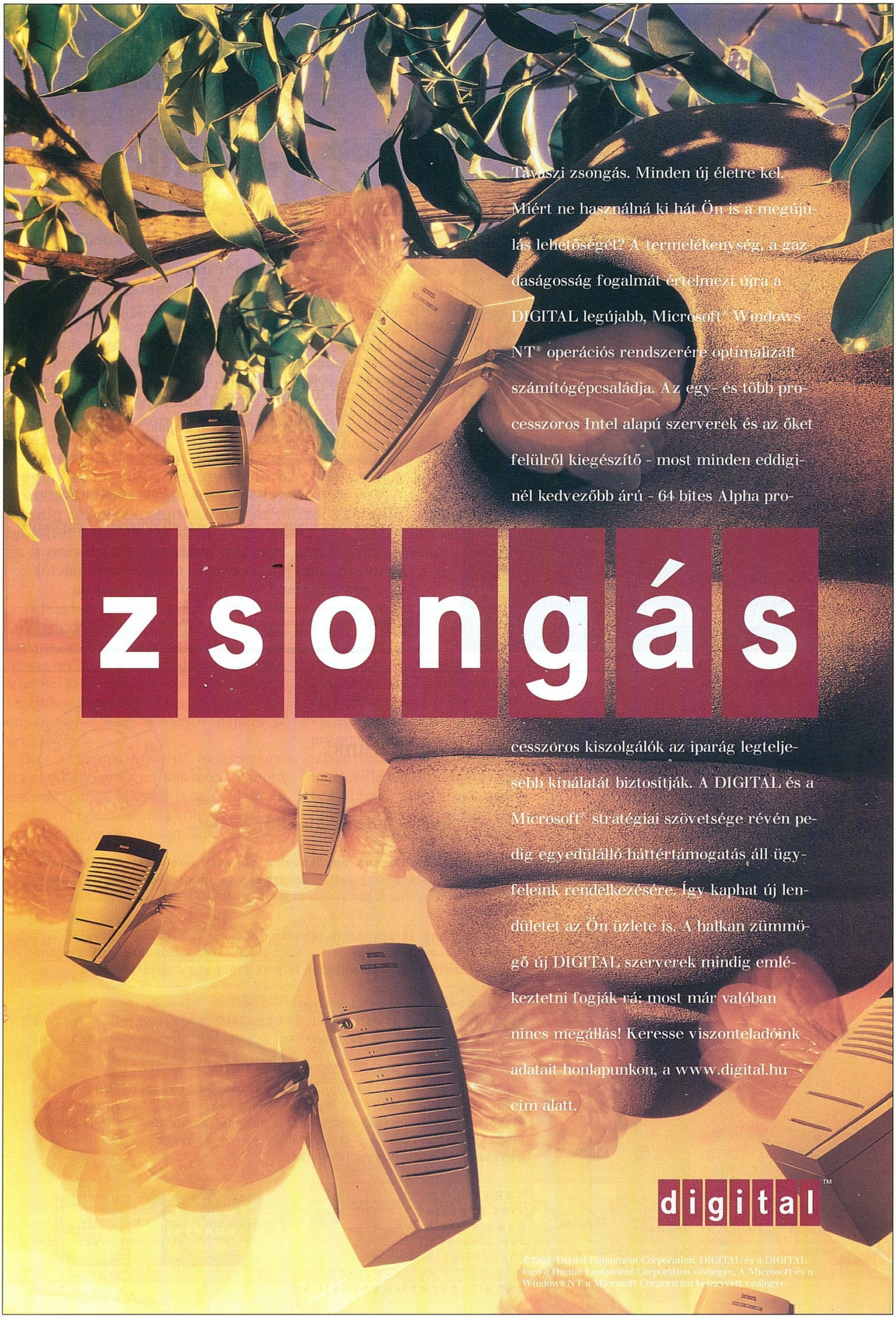
Elvileg a BIOS ki tud olvasni az EEPROM-ból egy verziószámot, és ezzel el tudja dönteni, melyik szabvány szerinti az SDRAM. Hogy azután mit tesz a régi verziójú SDRAM-mal, az már a BIOS programozójára van bízva. Vélhetőleg az új SPD szabvány már az egész világon elterjedt, így feltehető, hogy az új tervezésű SDRAM-ok korrekten vannak programozva. Ha a felhasználó egy LX chipkészletes alaplapot használ, tudnia kell, hogy csak az EEPROM-mal ellátott SDRAM-ok működnek korrekten.

Az LX chipkészlet

Az LX chipkészlet alapvetően csak a nem pufferelt 3,3 voltos RAM-ot támogatja. A DIMM modulokat két bevágással látják el, ezek pozicionálják a modult. Így nem fordulhat elő, hogy a 3,3 V-os modult az 5 V-os helyébe dugjuk, vagy fordítva. Az SDRAM-nak minimum 66 MHz-cel kell működnie. Az EDO RAM-nak legalább 60 ns-osnak kell lennie.

A chipkészlet 8 RAS vezetéke 4 memóriabank címzését teszi lehetővé, választhatóan egyoldalasan (1 RAS vezeték) vagy kétoldalasan (2 RAS vezeték). Az EDO és az SDRAM keverhető, ha a BIOS felismerő rutinja ezt lehetővé teszi. Némelyik OEM megengedi a keverést, a többi azonban nem. A felhasználónak javasoljuk: ha LX chipkészletes alaplapot használ, EDO RAM-ot csak akkor alkalmazzon, ha az OEM-gyártó ezt kifejezetten ajánlja.

Morva Sándor



INFORMÁCIÓKÉRÉS: 15 ▶

Photoshop suli — IV.

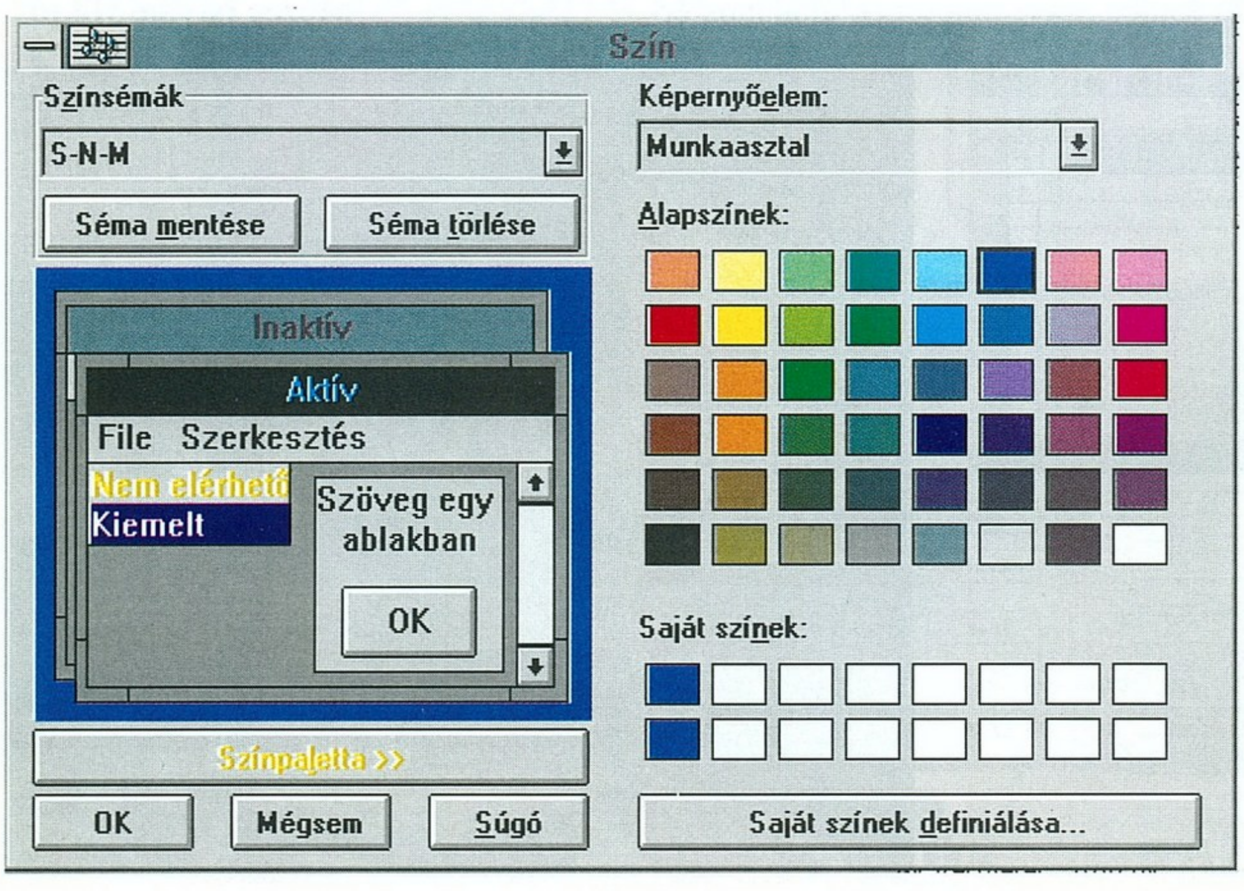
Munka a képekkel

Sorozatunk előző három része általános ismertetés volt (az első kettő a lapban, a harmadik a CD-n). A továbbiakban a képeken elvégzendő műveletekre fogunk összpontosítani, bemutatva az egyes képek feldolgozása során kínálkozó lehetőségeket. Egyszer az lesz a feladatunk, hogy egy meglévő képet egészében vagy részleteiben átszínezzünk, máskor az, hogy bizonyos képelemeket beillesszünk egy másik képbe, megint máskor az, hogy egy adott kép bizonyos részleteit áthelyezzük a képen belül, vagy egy kép részleteinek — esetleg egészének — bizonyos jellemzőit megváltoztassuk... és így tovább. Sorozatunkban a már tárgyalt ismeretekre mindig támaszkodunk, tehát az előző cikkekből fel lehet frissíteni a különböző eszközök használatával és a paraméterezésekkel kapcsolatos tudnivalókat.

A feladatokat zenei kifejezéssel élve „ujjgyakorlatoknak" szánjuk, de a Photoshop program lehetőségeiben az is benne rejlik, hogy akár alapvető műveletekkel kitűnő alkotásokat hozzunk létre. Ehhez persze türelem, idő és — ne tagadjuk — még bizonyos hajlam is szükséges.

Két általános érvényű megjegyzés. Az egyik, hogy a grafikus munka finom egérmozgatást követel meg tőlünk, és a gyakori egértisztítás elmulasztását kiváló kézügyességgel sem lehet ellensúlyozni.

A másik ajánlás a képernyőre vonatkozik. A Windows alapértelmezés szerinti fehér háttere nemigen teszi lehetővé a képeken szereplő megvilágítási különbségek és a színárnyalatok pontos érzékelését, ezért javasoljuk, hogy a vezérlőpulton a színek beállításánál a program munkaterületét állítsuk be sötétszürkére, az ablak hátterét pedig semleges világosszürkére. (1. kép.) Ezzel a beállítással más programokban is jól dolgozhatunk, és a szemünk sem fog elfáradni. (Ráadásul a képernyőt is ez kíméli a legjobban.)



1. kép

1. lecke: képrészlet átszínezése

Az a feladat, hogy az mellékelt képen szereplő modell hajának színét változtassuk meg, mégpedig a frizura megtartásával. Ezt többféle módon is megtehetjük.

a) Először csináljuk meg a módosítást az Image legördülő menü Adjust/Color Balance parancsával.

1. Hozzuk létre a kép másolatát egy új rétegen a Duplicat Layer paranccsal. Az új réteg legyen átlátszatlan (Opacity = 100%). A továbbiakban ezt a lépést tekintsük magától értetődőnek, amikor egy változtatási feladathoz hozzákezdünk.

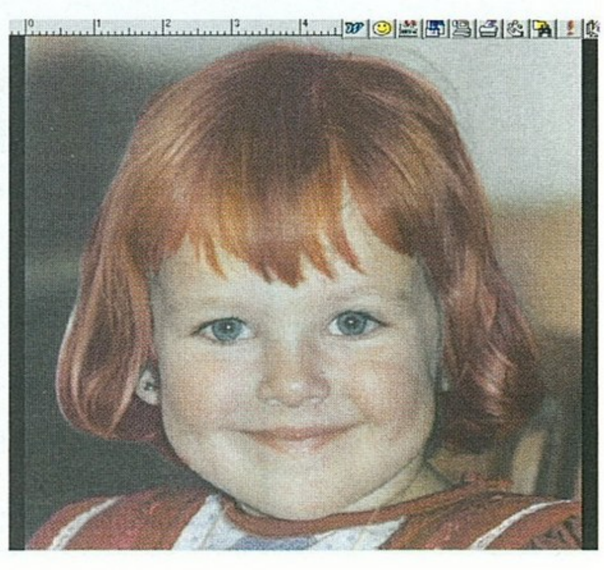
2. Jelöljük ki a modell haját a lasszó eszközzel. Megfelelő nagyításnál egészen pontosan dolgozhatunk, a Tab billentyűvel eltüntethetjük a zavaró palettákat. (2. kép.)



2. kép

3. Most adjuk ki a fent említett parancsot. A csúszkák segítségével állítsuk be a kívánt színárnyalatot. A mellékelt képen ez vörös, de természetesen bármilyen színt kikeverhetünk. (3. kép).

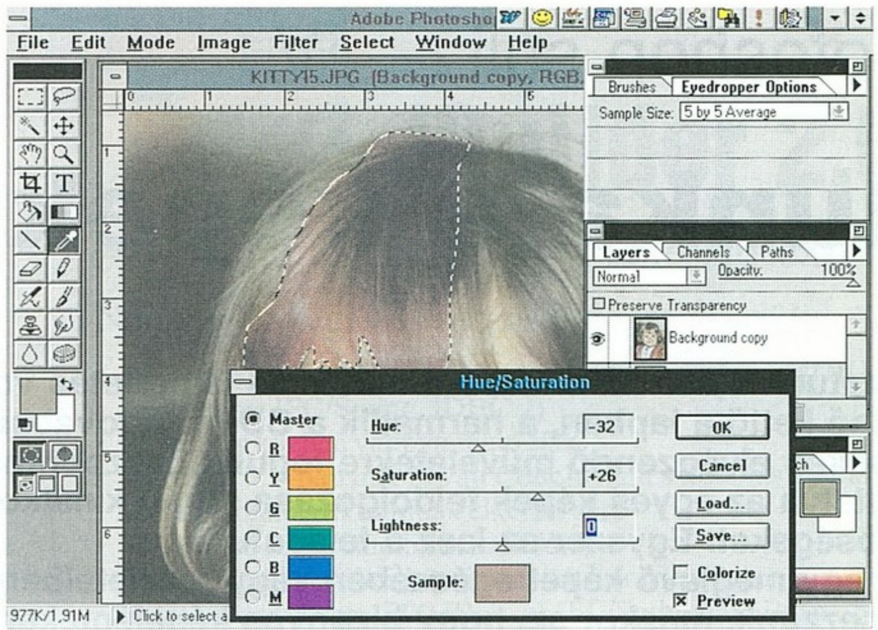
4. Ha elégedettek vagyunk az eredménnyel, a Flatten image paranccsal (Layers paletta) olvasszuk össze egyetlen képpé a rétegeket.



3. kép



4. kép



5. kép

A kiválasztás pontosságának megfelelően jó eredményt érhetünk el. Annyi történt csak, hogy a kiválasztott terület pixeleinek színegyensúlyát megváltoztattuk.

b) Más módon is megközelíthetjük a feladatot.

1. Jelöljük ki a területet.

2. Az Image/Adjust/Hue/Saturation parancs kiadása után a három csúszka segítségével állítsuk be a kívánt szín árnyalatát, telítettségét, és ha szükséges, a fényességét. Ezt megtehetjük a Master opció bejelölésével — ekkor az összes színösszetevő módosul —, de kiválaszthatunk egy adott színt is. (4. és 5. kép.)

A Hue/Saturation segítségével aprólékosabb, de bonyolultabb változtatásokat eszközölhetünk, mint az előbbi módszerrel.

c) Most nézzünk meg egy harmadik módszert.

1. A kijelölt területre adjuk ki az Image legördülő menüből az Adjust/Selective Color parancsot.

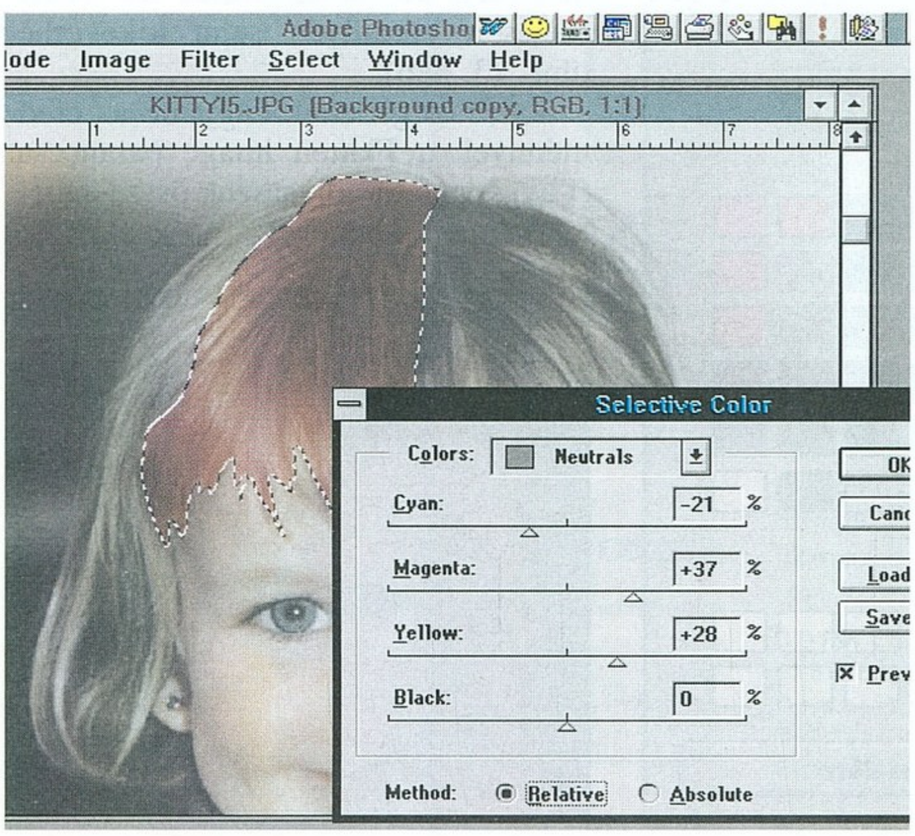
2. Nézzük meg, hogy esetünkben melyik színt érdemes módosítani. (Amint azt korábban leírtuk, az úgynevezett feketepróbával egyértelműen látni fogjuk, hogy az adott színből mennyit tartalmaz a kiválasztott terület, tehát mennyire érdemes valamelyik színhez hozzányúlni.) Mostani képünk esetében a haj színe erősen kevert szín, ezért itt a leglátványosabb eredményt a neutrális színek megváltoztatásával értük el. (6. és 7. kép.)

Példáink mindegyikében vörösre színeztük a modell eredetileg középszőke haját. Természetesen más színekből kiindulva is ezekhez hasonló eredményre juthatunk, vagy akár extrém színezeteket is produkálhatunk.

Sok más módszer is használható a színmódosításokra, ízelítőnek talán ennyi is elég. Mindhárom módszer esetében ajánlatos a különböző beállítási módszerekkel kísérletezgetni, ami azért is fontos, mert nincsenek általános érvényű instrukciók arra nézve, hogy milyen beállításokkal hozhatjuk létre a kívánt módosításokat, hiszen a különböző képek színe, telítettsége és fényessége több paraméter szinte végtelenül sok kombinációjából állhat össze, még ha ránézésre sok is a hasonlóság. Kellő gyakorlattal azonban már tudni fogjuk, hogy milyen módszerrel érdemes közelíteni az adott feladat megoldásához.

Gyakorló feladat: Próbáljuk ki egy kép kiválasztott területén az Image/Adjust/Variations parancsot hasonló eredmény elérésére. A színes fotók kidolgozásában járatos felhasználók itt otthonosan fogják érezni magukat, és bizonyára megkedvelik ezt a metódust, amelynek segítségével nyersanyag pocsékolása nélkül, gyorsan és egyszerűen lehet korrigálni számos felvételi hibát — akár még olyanokat is, hogy napfényfilmre korrekciós szűrő nélkül exponáltunk műfényben, vagy fordítva.

Mózes István Miklós



6. kép



7. kép

VHDL — a jövő útja a tervezésben?

Hardverleíró világnyelv

A hardverfejlesztők két fő csoportját különböztethetjük meg: az integrált áramkörök tervezőit és a rendszertervezőket. Mind a két csoportnak pontosan meghatározható a feladata: az IC-tervezők olyan alkatrészeket készítenek, amelyeket majd a rendszertervezők használnak a kártyák és az ezekből álló berendezések megépítéséhez. Ez a „két világ" sok mindenben különbözik, így a tervezési módszerekben, eljárásokban is. Felvetődik a kérdés: hogyan lehetne közelebb hozni egymáshoz e kettőt, megkönnyítve a tervezési munkát? Egyre inkább úgy tűnik, hogy a választ a leíró nyelveknél kell keresnünk.

A mikroelektronikában nagy szerep jut az alkatrészek tesztelésének, mert ezen a területen kerül a legtöbbe az elkövetett hibák korrigálása. A szimuláció ugyan nem tesz lehetővé kimerítő tesztelést, de segítségével a hibák nagy százalékát diagnosztizálhatjuk.

A technológiai módszerek fejlődésével az IC-k komplexitása is növekszik. Nagyon leegyszerűsítve azt mondhatnánk, hogy amihez pár évvel ezelőtt még egy egész rendszerre volt szükség, arra most egyetlen integrált áramkör elegendő.

Ennek következtében viszont mind nehezebbé, sőt szinte lehetetlenné válik a rendszer globális szimulációja. Természetesnek mondható tehát, hogy a mikroelektronikában megjelentek a hardverleíró nyelvek, amelyek nagymértékben megkönnyítették a szimulációt. (Nem is a VHDL volt az első.) Ilyen módon a tervezési idő lerövidült, ami óriási előny a piaci versenyben.

A leíró nyelveknek más szerepük is van: talán ők alkothatják azt a bizonyos interfészt a fentebb említett „két világ", az IC-tervezők és a rendszertervezők között. A VHDL nyelven „elkészített" alkatrészek lehetővé teszik a teljes rendszer tesztelését, és a rendszer egy részének a környezetétől független vizsgálatát. A hibák kijavítása is sokkal egyszerűbbé válik. Az IC-gyártók számára, akiknek legnagyobb vetélytársuk az idő, lehetővé válna, hogy megrendelőiknek egy VHDL-ben (is) megvalósított alkatrészt adjanak át. A megrendelők pedig ugyanazon a VHDL platformon integrálhatnák rendszerükbe a sokféle alkatrészt. Ezek a lehetőségek Európában egyelőre többé-kevésbé még csak elképzelések, de a tengerentúlon már valóságosak.

Normalizálásával a VHDL nyelv megbízható, mindenki számára könnyen megismerhető, lehetőséget ad a szimulációra, az ellenőrzésre. Alkalmazási köre igen nagy, arra is volt példa, hogy mechanikai rendszerek tesztelésére használták. A leíró nyelvek előnyeit korábban is felismerték a tervezők, de azok megalkotásához komoly beruházásokra, sok munkára volt szükség. Cserébe viszont jó hatásfokú, könnyen használható, a céloknak megfelelő nyelvek születtek. Volt azonban egy nagy hátrányuk: ahhoz a fejlesztési környezethez kötődtek, amelyhez a tervező a nyelvet megírta, nyilván saját munkájának megkönnyítésére. Ezzel szemben a VHDL talán legnagyobb előnye, hogy IEEE standard, így a többi nyelv közötti fordítók helyett egyszerűbb megalkotni a VHDL felé az interfészt, ezáltal a fejlesztők egymás specifikációit, terveit, leírásait munkájuk során felhasználják.

A VHDL hardverleíró nyelv részletes bemutatására itt nincs lehetőség. Aki szeretné jobban megismerni ezt a világot, bővebb információkat is csak az idegen nyelvű szakirodalomban találhat. (Magyar nyelvű kiadás tudomásunk szerint még nem készült erről a szakterületről.)

A hardverleíró nyelvek nagy részének elnevezésében szerepel a HDL betűszó: Hardware Description Language. A V betű egy másik betűszóból származtatható: VHSIC = Very High Speed Integrated Circuits. (Nem teljesen tisztázott, hogy a „H" betű a HDL-ből, vagy pedig a VHSIC-ből ered.) Az elnevezésből arra következtethetnénk, hogy a nyelv csak a mikroelektronika területén használható, de a VHDL az integrált áramköröknél jóval magasabb absztrakciós szintű leírást is lehetővé tesz: kártyák, sőt egész rendszerek hardver- és szoftverrészének modellezésére is szolgál — tehát algoritmusokat is tartalmaz. Ez a sokoldalúság teszi a nyelvet igazán jelentőssé.

Az igény a rendszerek egyértelmű és általános (standardizált) leírására az Amerikai Védelmi Minisztériumban vetődött fel először a 80-as évek elején. Az Intermetrics, az IBM és a Texas Instruments 4 éves munkájának eredményeképpen született meg az IEEE 1076B szabvány, és azt 1987. december 10-én hagyták jóvá. Azóta a VHDL nyelv a VASG (VHDL Analysis and Standardisation Group) felügyelete alá tartozik, ami a DASS (Design Automation Standard Subcommittees) egyik alcsoportja, és ilyen módon IEEE szervezet.

A hardverleíró nyelv az általános programnyelvektől leginkább talán alap-építőegységeit tekintve különbözik. A programozási nyelveknél ezek az alapelemek a szubrutinok és az alprog-

ramok, a leíró nyelveknél viszont a komponensek (component). Egy szubrutint egyszer meghívunk, az teszi a dolgát, majd el is feledkezhetünk róla, egészen addig, amíg újból meg nem hívjuk. Egy komponens viszont, ha egyszer megírták, időtől függetlenül létezik: a rendszer komponensei statikus módon „élnek", és egymással párhuzamosan működnek. Így tehát az időtényező a leíró nyelvekben nem játszik szerepet — pontosabban szerepe csak az adatáramlás szintjén, az adatok ki- és bevitelekor van.

Egy másik fontos különbség a jelek megjelenése a leíró nyelveknél. Egy programnyelvben csak változókról beszélhetünk, és a változó élettartamát behatárolja a szubrutiné, amelyikben deklarálva lett, míg egy jel (a VHDL nyelvben) a szimuláció egész időtartama alatt ott van, és rendelkezik múlttal — hogy mikor, milyen értékeket vett fel eddig — és előre látható jövővel (pilot = melyik időpillanatban milyen értékeket kell felvennie). Összefoglalva, ha egy változónak értéket adunk, az időben limitált esemény lesz:

A :=B

egy egyszeri történés, mégpedig hogy A felveszi B értékét. Ezzel szemben, ha egy jel értékét határozzuk meg:

A <= B
B <= C

az a jelek permanens összekapcsolását, rövidre zárását jelenti.

Itt kell megemlítenünk a konfliktusfeloldó függvények szerepét a VHDL nyelvben. Az előző példa kapcsán, ha mondjuk B-re és C-re programunkból valamilyen okból különböző értékeket kényszerítünk, akkor konfliktus jön létre, mert a két jel rövidre van zárva. Ebben az esetben a VHDL meghívja a konfliktusfeloldó függvényt, amelyben meg van adva, hogy konfliktus esetén hogyan kell eljárni. Ha nincs ilyen függvény, és fellép a konfliktus, a szimulátor hibajelzést fog adni. Ha nem írtunk konfliktusfeloldó függvényt, de konfliktus sem lépett fel, viszont megvan rá a lehetőség, az eredmény csak egy várakozás (warning) lesz.

Amint már szó esett róla, a VHDL IEEE szabvány, a tervezési eszközöket fejlesztők által általánosan elismert standard. A gyártók tehát nagyobb hajlandósággal fognak a nyelvbe beruházni, mivel a normalizálásnak köszönhetően hosszú távra tervezhetnek, csökkentve a kockázati tényezőt. Egy tervező nem szívesen fog majd olyan alkatrészt integrálni a rendszerébe, amelynek nincs VHDL modellje, mert akkor nem tud a rendszer egészén szimulációt végezni. Ezeket az elvárásokat a leíró nyelvek közül csak a VHDL tudja teljesíteni, a többi nyelv vagy nem képes felvenni vele a versenyt, vagy pedig inkább fejlesztettek hozzá egy „VHDL-interfészt".

Az Amerikai Védelmi Minisztérium kötelezően elvárja szállítóitól, hogy a megrendelt IC-kről, de ma már a kártyákról, sőt a teljes rendszerekről is adjanak VHDL leírást. E leírások egyébként ma már katonai szabványok is (MIL STD 454 requirement 64). Ezt az álláspontot az európai gyártók sem hagyhatják figyelmen kívül, ha nem akarják a világpiac jelentős részéből kirekeszteni magukat.

A VHDL nyelv technikailag is igen modern és hatékony. A benne készített leírások könnyen átláthatóak, jól megérthetőek, megbízhatóak, és nagyfokú modularitással rendelkeznek. Mindez olyan fogalmaknak köszönhető, mint a konfliktusfeloldó függvény, az önmagukban fordítható részegység, az idő egyértelmű definiálása stb. Egy leíró nyelv segítségével készült termék gyártása során a megoldandó probléma elemzése után a megvalósítás (jelen esetben a kód megírása) szakaszához tartozik a tesztelés és a hibák kijavítása is. Ez utóbbi természetesen annál többe kerül, minél későbbi fázisban kerül rá sor. Erre való tekintettel a modern nyelveket — így a VHDL-t is — úgy próbálták megírni, hogy a hibák döntő többsége már fordításkor jelentkezzen. Ugyancsak pénzügyi szemszögből előnyös a nyelv hierarchikus és moduláris felépítése. Ennek köszönhetően ugyanis minden nehézség nélkül megváltoztatható egy termék vagy egy termékkomponens, de még a specifikáció is, a tervezés akármilyen késői fázisában! (Ha belegondolunk, ez persze nem biztos, hogy feltétlenül előnyös...)

Ennyi pozitív tulajdonsága mellett ejtsünk szót a hardverleíró nyelv hátrányairól is. Az igen hatékony VDHL komplexitása is elég nagy. Rendszertervezők számára készült, akik nem feltétlenül rendelkeznek programozási gyakorlattal, és azt esetleg a VHDL nyelv tanulmányozása folyamán kell elsajátítaniuk.

A nyelvnek vannak hiányosságai a szimuláció területén is, de ezek az újabb szimulátorszoftverek megjelenésével valószínűleg fokozatosan el fognak tűnni.

Zsadányi Pál

Tartalomra várva

Az új média születése

1997 tavaszán a New York-i Columbia Egyetem Informatikai Intézete konferenciát tartott annak megvitatására, hogy a számítástechnikai és kommunikációs modellek fejlődésével párhuzamosan merre tart a sajtó, és mi várható a közeljövőben. A konferencián több száz szerkesztő, író, kiadó, informatikai szakember, kutató, producer és egyetemi tanár vett részt, de nem tudtak válaszolni arra a kérdésre, hogy kialakulóban van-e, illetve ki fog-e alakulni egyáltalán az ún. „új média".

A problémák ott kezdődnek, hogy nehéz meghatározni, mi is lenne ez, hiszen olyasvalamiről beszélünk, ami a valóságban még nem létezik. A legtöbb kutató azonban egyetért abban, hogy bár a technológia fejlődése nagymértékben hatott a médiára, a formaváltással nem járt együtt a tartalom lényegi átalakulása. Márpedig igazán „új média" az lenne, ha az új hordozók új tartalmat is közvetítenének.

Nagyon messzire vezetne, ha megpróbálnánk pontosan meghatározni ezt az új tartalmat, de ha belegondolunk: a műholdas technológia jóvoltából a 90-es évekre lehetségessé vált, hogy Közép-Afrikában fogják valamelyik Turner-kábelcsatorna műsorát, miközben ez még mindig az 50-es, 60-as években készült szappanoperákkal van tömve... — nos, akkor sejthetjük, miről is van szó. Az pedig talán ennél is messzebbre mutató példa, hogy az írott sajtó elektronikusan elérhető termékeinek túlnyomóan nagy százaléka nem áll másból, mint html-esített szövegből. A tartalmat rendszerint nem alakítják át úgy, hogy az visszatükrözze a befogadó technika, a hipertext lehetőségeit.

Kérdés tehát, hogy a számítástechnika eszközrendszere és kultúrája a hatalmas sebességű fejlődés ellenére miért nem gyakorolt döntő hatást a tömegtájékoztatás médiáira — különösen pedig az írott és sugárzott sajtóra.

Jay Bobowitz, a Hearst sajtóbirodalom „New Media and Technology" csoportjának alelnöke alapvető problémára mutatott rá, amikor megállapította, hogy „az elektronikus forradalom kezdete óta a technológia ül a kormánynál és határozza meg az utat, a hírközlés tartalma pedig [csupán] megpróbálja azt követni". A technológia által létrehozott új formák és nem kevésbé új hordozók (az Internet, a World Wide Web, a multimédiás alkalmazások, a hipertext stb.) egyelőre gyakorlatilag nincsenek hatással a tartalomra — azaz a sajtó továbbra is a régi dolgokkal akarja megtölteni az új formákat.

Lássuk hát, miként is jött létre ez a helyzet, és milyen változások várhatóak.

Egy jól ismert tézis szerint az, amit manapság information societynak szokás nevezni (tehát az a társadalom, amelyben a régi mezőgazdasági, majd ipari termelési alapokról az „informatikai termelésre" helyeződik át a hangsúly, és a munkavállalók nagy része az információs szektorban dolgozik) az utóbbi évtizedek bontakozott ki. A folyamat kezdetét különféle időpontokra szokás helyezni, de abban a legtöbben egyetértenek, hogy a fő ok a komputertechnológiák és a kommunikációs technológiák konvergenciája (egymáshoz közelítése) volt.

Ennek eredményeként a számítástechnika nem maradt a katonai vagy egyetemi laboratóriumok falai között, hanem részévé vált a mindennapi életnek is (ezért teszi például *Steve Wozniak,* az Apple Computers egyik alapítója akkorra az új korszak kezdetét, amikor megszületik a személyi számítógép), a hírközlési és kommunikációs technikák pedig ugrásszerű fejlődésnek indultak, hogy szükségszerű összeolvadásukból megszülessen az az információs társadalom, amelyben ezek használata nemcsak mindennapossá, de bizonyos értelemben létszükségletté is vált (gondoljunk csak arra, hogy a szövegszerkesztő és az e-mail ismerete nélkül ma már egy titkárnő el sem tud helyezkedni).

Nyilvánvalónak látszik, hogy a technika irányítja a fejlődést, és hasonlóképpen nyilvánvalónak tűnik az is, hogy ennek komoly hatása lesz a társadalomra meg a mindennapi életre egyaránt. Nehéz azonban megállapítani, hogy erre a technika által életre hívott új társadalomra mi is lesz jellemző valójában — nem véletlen, hogy ezt a szociológusok nem önmagában, hanem inkább valamihez képest határozzák meg. Azaz amikor arról beszélnek, hogy az ipari szektorról a szolgáltatóiparra tolódik át a hangsúly, akkor a „posztindusztriális" társadalom fogalmát használják.

Egyfelől tehát a technológiai változások hatnak a társadalomra, másfelől pedig az átalakulóban lévő szocio-kulturális struktúrák (nevezzük őket posztindusztriálisnak) olyan esztétikai, kulturális kategóriát hoztak lére, amit „posztmodern" néven szokás emlegetni. Erről azért érdemes kissé bővebben is beszélni, mert ez (vagyis a „posztmodern fordulat") az a másik tényező, amely döntő befolyással van az új média kialakulására.

Mindemellett bizonyos szempontból persze kockázatos lehet posztmodern médiát emlegetni, hiszen a „modern" média sem alakult ki mindenhol. Főleg nem a televízió esetében, amire éppen a berögződött, változatlanná merevedett műfajok (szappanoperák stb.) lettek jellemzőek. Persze az is igaz, hogy az MTV vagy a Max Headroom sorozat szinte új műfajokat alkotott, és ezeket a szakértők posztmodernnek szokták tekinteni.

Ráadásul a médiák fejlődési történetében mindenhol ki lehet mutatni egy sokszor „képi fordulatnak" is nevezett „posztmodern fordulatot". Itt nagyjából arról van szó, hogy a korábbi, szövegalapú kommunikációt felváltja a kép uralma, illetve olyan esetekben (például a televíziónál), ahol mindig is a képi formák voltak meghatározóak, az egyenes vonalú és megszokott, narratív történetmesélést fölváltja a posztmodern montázs. A képi megjelenítés mélységét veszti, felszínessé és széthullóvá válik, felgyorsul — és az egész koncepció egy, a korábbitól gyökeresen eltérő normához próbál meg igazodni.

A televízió esetében ez a fordulat a már említett műsorokkal kezdődött, illetve a televíziós híreknél szintén a nyolcvanas évek közepén, mikor az új vágási technikák és képalkalmazási fi-

lozófiák nyomán kialakult az a „tabloid television", ami már nemcsak tartalmában volt „bulvártévé", hanem technikáit tekintve is.

Az írott sajtónál a változás a hordozó közeg korlátai miatt kevésbé volt észlelhető, de ha erről a témáról van szó, akkor a USA Today-t kell említenünk, mert elsőként ez alkalmazott olyan mennyiségű vizuális szemléltető anyagot, hogy az túlsúlyba került a szöveggel szemben.

Ami pedig az informatikát és a számítástechnika által lehetővé tett tömegkommunikációt illeti, itt a „képi fordulat" egyértelműen egybeesett a Web forradalmával: a grafikus böngészők meg a képek megjelenésével. A mostani vizuális kultúra dominanciáját egyébként mi sem érzékelteti jobban, mint ez a változás. Az Internet a Mosaic meg a hipertext előtt csak a beavatottak unalmas és nehezen élvezhető játszótere volt: a publikum csak akkor merészkedett be ide, amikor a grafikus felhasználói felületek elterjedésével megjelentek az ismerős és megnyugtató képek (természetesen ugyanez igaz az Apple komputerekre, és ez magyarázza meg azt is, hogy miért létezik ma Windows).

Mindenesetre a grafikus felület megjelenésével a Web csatlakozhatott a képi forradalomhoz, a web-publishing pedig szinte egyenjogú félként az eddig uralkodó médiákhoz (sajtó, rádió, tévé, film stb.). Ezzel a folyamattal párhuzamosan viszont az írott sajtó emberei kezdtek egyre hátrányosabb helyzetbe kerülni.

Tisztában voltak ugyan vele, hogy a web-technológia elterjedésével radikális változások várhatók, mégsem tudták, hogy mit tegyenek. A legradikálisabb fordulatra, a totális online-osításra nem voltak hajlandók, hiszen az a hagyományos forma végét jelentette volna. Ehelyett megpróbáltak a menet élén maradva fél szemmel álandóan az Internetre sandítani. Kb. három évvel ezelőtt, amikor nyilvánvalóvá vált, hogy a Web meghatározó tényező, több tekintélyes lap — élükön a Time magazinnal — állandó Internet-rovatokat indított, valamint egyre több internetes információt közölt.

Ezek a próbálkozások azonban legfeljebb szánalmasnak nevezhetők: az Internet sajátosságaiból fakadóan erről a témáról élvezhető újságcikkeket írni gyakorlatilag lehetetlen. A vállalkozás zsákutcának bizonyult, amelybe nem is sikerült becsalogatni az olvasókat. Azon buzgólkodván, hogy bebizonyítsák, az Internet leginkább pornográf anyagok soha nem látott méretű hordozója, a Time magazin szerkesztői hatalmas öngólt lőttek, és csökkent a szakmai presztízsük. (Dióhéjban az történt, hogy hatalmas címlapsztorit közöltek, amelyben a Carnegie Mellon Egyetem éppen akkoriban napvilágot látott tanulmányára hivatkozva azt állították, hogy a világhálózaton megtalálható anyagok vagy pornográfok, vagy más szempontból, de mindenképpen elfogadhatatlanok a társadalmi normák szempontjából. Rövidesen azonban kiderült, hogy az állítások alapjául szolgáló felmérés nevetségesen inkompetens volt — például nem publikus Internet-helyeket vizsgált, hanem privát BBS-eket —, és szerzője az évek során több hasonló botrányt próbált már kavarni. A Timenak mindent vissza kellett vonnia, és sokan repültek a laptól.)

Mindent egybevetve az írott sajtóról tehát azt mondhatjuk, hogy komoly nehézségekkel kell szembenéznie: vagy szakít a hagyományos formákkal meg a lineáris elbeszélő technikával, hogy a szövegalapú tartalmat megfelelő képi oldatban, interaktív formában találják fel, vagy izolálódik, és elveszíti jelentőségét.

A televízió korántsem adta fel a játszmát — és nem adta fel azt a reményt sem, hogy felhasználja az új kommunikációs technológiákat. Gondoljunk csak az MSNBC-re, ami, miként a neve is sejteti, az NBC hagyományosabb „content"-jét a Microsoft technológiájával akarja közvetíteni (bár itt természetesen már a számítástechnika maga egy bizonyos mértékig „hírré" válik). Ugyanakkor az MSNBC nélkülözi a Web legfontosabb ismérvét, az interaktivitást. A néző éppúgy néző marad, mint eddig, és nincs különösebb beleszólása abba, hogy mi kerül a képernyőjére.

Éppen ezt a helyzetet kívánja gyökeresen megváltoztatni *Bill Gates* egyik legújabb kísérlete. A Microsoft Corporation 1997 végén jelentette be, hogy piacra dobja a WebTV-t. A koncepció már évek óta létezik, működő prototípusai, sőt, kereskedelmi forgalomban lévő változatai is vannak már, eddig azonban súlyos kudarcok kísérték pályafutását.

A mostani kísérlet az eddigieknél már csak azért is figyelemreméltóbb, mert mögötte a Microsoft és a Sony Corporation dollármilliárdjai állnak. Ami lehet, hogy nem éppen tudományos szempont, de semmiképpen nem elhanyagolható.

A WebTV volna az első komoly kísérlet a különböző kommunikációs eszközök összehozására: jelen esetben a televízió és a World Wide Web összeházasítására. Az eredmény pedig egy olyan doboz, amely mindennel fel van szerelve: egérrel, billentyűzettel és távirányítóval is. Benne pedig egyszerre lehet végezni a posztmodern tevékenységek két legalapvetőbbikét: a szörfözést a tévécsatornákon és a weblapokon.

Ezen a ponton óhatatlanul felmerül a kérdés, hogy akkor ez lenne-e az? Márminthogy a WebTV-vel megérkezünk-e már a sokat emlegetett New Media birodalmába?

Én személy szerint ezt igencsak kétlem. Bár naivitás volna Bill Gates dollárkapacitása ellen fogadni, azért az is fontos szempont, hogy a WebTV nem válasz az elején felvetett kérdésre, nevezetesen arra, hogy csatlakozik-e új tartalom is a technikai forradalomhoz, illetve az esztétikai-kulturális változások által olyannyira átalakított kommunikációs formarendszerhez, vagy minden marad a régiben, és csak a csomagolás változik.

Úgy tűnik, hogy a WebTV (vagy például az az Amerikában mostanában elterjedő minidish rendszer, amely szintén egyetlen dobozba akarja belekényszeríteni a számítógépet, a kábeltelevíziót, a telefont és az Internetet) közönséges érdekházasság csupán, ahol a kényelmi szempontokon túl más nemigen érvényesül. Mert attól valóban kényelmesebb lehet ugyan a Web meg a tévé használata, hogy egy dobozban vannak, de ez még semmiféle új tartalmat nem jelent.

Jelenleg meglehetősen nehéz jóslatokba bocsátkozni azzal kapcsolatban, hogy megszületik-e végül a várva várt új média, vagy az új formákat továbbra is régi tartalom fogja megtölteni. A késlekedés számtalan okra vezethető vissza — kezdve azon, hogy az információközlési struktúrák sokkal lassabban változnak, mint hordozóik (egy száz évvel ezelőtt kiadott újsággal például még egészen jól elboldogulunk, egy kétszáz éves lap már viszont szinte érthetetlen számunkra), egészen addig bezárólag, hogy az új tartalmat, ha szabad így fogalmaznunk, valakinek le is kell gyártania. Márpedig az ilyen vállalkozások áthághatatlan akadályokba fognak ütközni mindaddig, míg a mostani média- és szórakoztatóipar urait és irányítóit nem sikerül megnyerni a dolognak.

Komáromy Gábor

Ez az írás először előadásként hangzott el, 1997. december 12-én az OMFB támogatásával megrendezett „Számítógépes kultúra: kihívás és fenyegetés" című konferencián, a Budapesti Műszaki Egyetemen.

Élt kerek 8 évet...

Floppynekrológ

A címbeli tömörítésnél egy kicsit pontosabban fogalmazva, még nem magáról a floppyról mint adathordozóról kell nekrológot írni (egyszer bizonyára annak is eljön az ideje), hanem az [Új] Alaplap floppymellékletéről, amely mostani számunkban jelenik meg utoljára, hogy helyét teljeskörűen átadja a CD-nek. Ezzel lezárul egy nyolcéves korszak. Nem érezzük úgy, hogy ebből az alkalomból valami nagyon szomorú búcsúztatót kellene írni, de a visszaemlékezés egy kicsit ezt is előcsalja belőlünk.

Nem mindenki tudja, hogy mielőtt a Mikroszámítógép Magazin 1990. júniusában Alaplap névvel és floppymelléklettel megjelent volna, 1989-ben már létezett az Alaplap című, önálló „mágneslemez-újság", és azt az akkori Cédrus Kisszövetkezet újszerű reklámhordozónak szánta, a sorozatban elindított „floppy-lapok" egyikének.

Azt talán még kevesebben tudják, hogy az állandó floppymelléklet bevezetésének volt egy speciális indítéka is. A Mikroszámítógép Magazin rendszeresen közölt forráslistákat, és igen sok kellemetlenség származott abból, hogy gyakran voltak benne elütések, sajtóhibák. Ha a forráskódot eredeti digitális formában közöljük, akkor egyrészt a hibákért már nem a szerkesztőséget szidják, hanem a szerzőt, másrészt ha a kódot lefordítják, és a program fut, akkor a forráskód „szövegileg" jó. Nem biztos, hogy programhibáktól is mentes, de sajtóhibáktól valószínűleg igen.

Komplett lap a lapban

A lemezmelléklet forráskódközlő funkcióját mindvégig megtartotta, igazi vonzerejét később mégis inkább a hasznos kis programok közreadása jelentette. Egyrészt a világon bárhol felbukkant és jónak talált public domain, freeware és shareware szoftverekből adtunk közre tesztelt és gondosan válogatott programokat (a sokáig 360 Kbájtos floppy szűkös helye eleve igényelte is az alapos szelektálást), másrészt kezdettől fogva felkaroltuk a magyar programozók szárnypróbálgatásait, és helyet adtunk alkotásaik nyilvános megmérettetésének.

Az Alaplap lemezmellékletének „privilégiumát" az első néhány évben a többi számítástechnikai lap nem is próbálta megtörni, amikor pedig mégis felfedezték ennek vonzerejét, akkor is inkább csak rálapátolták az anyagot a floppyra. Az [Új] Alaplap mágneslemez-melléklete ezzel szemben „utolsó lehelletéig" önálló elektronikus újságként jelent meg. Ezt a különlegességet tetéztük idén egy még különösebb megoldással (ami talán egyedülálló a műfaj történetében), hogy 5 hónapon keresztül a nyomtatott lap futtatható CD-mellékletére rátettük az arról ugyancsak önállóan futtatható floppymellékletet. (Vagyis a papírújság CD-újságjába egy floppyújság is „be volt csomagolva".) Ennek a kuriózumnak azonban a következő számtól kezdve vége.

Az 1997 áprilisában lebonyolított olvasói közvéleménykutatás alapján még arra lehetett következtetni, hogy akár többszázan is lennének olyanok, akik környezetében nincs (sem otthon, sem munkahelyükön, sem baráti körükben) CD-meghajtóval felszerelt gép, ezért előfizetnének a floppys változatra, ha maradna ilyen lehetőség. Teremtettünk ilyen lehetőséget. Egy év alatt azonban a jelek szerint annyira általánossá vált a CD-használat, hogy az 1998 januárjától a floppys előfizetés csak az összes megrendelés 2-3 százalékát tette ki, száznál is kevesebb előfizetőnek külön floppymellékletet készíteni viszont értelmetlenül gazdaságtalan feladat.

Hagyományőrzés a CD-n

A floppys előfizetőket levélben már értesítettük a változásról, így idejében eldönthetik, hogy körülményeiktől függően előfizetésük módosításának melyik lehetőségét választják. A technikai trendek ismeretében elsősorban a CD-olvasás megoldását javasoltuk mindenkinek, mert egyelőre a floppy mellett, később pedig valószínűleg helyette is a CD és a DVD különféle változatai lesznek a fő mozgatható, írható és újraírható adathordozók. Szeretjük vagy nem szeretjük ezt a megalomániás fejlődési vonalat, változtatni nem tudunk rajta.

Szűk kis szerkesztőségi csapatunk még vállalni tudta a floppymelléklet készítésének teljes vertikumát, beleértve nemcsak az anyaggyűjtést, a szerkesztést, a floppy működő újsággá történő „összevarrását", hanem 10 000 feletti példányszámban a kisipari gyártást és címkézést is. 1998 januárjától sok minden megváltozott, mellékletkészítésünk jóval költségesebb lett. A CD-melléklet legyártása ekkora mennyiségben a jelenlegi technológiai adottságok közepette eleve csak nagyipari körülmények között történhet. A 650 megabájtnyi anyag válogatását, tesztelését, szerkesztését sem lehet már a nyomtatott lap mellett úgy megoldani, ahogy 1,44 megabájtos floppy esetében az még sikerült. Ahhoz, hogy floppymellékletünk hírnevéhez méltó CD-mellékletet tudjunk produkálni, külső erőket kellett a feladattal megbíznunk, tehetséges csapatot választottunk, de a CD-n még bőven van csiszolnivaló...

A CD-koncepció kialakítása mellett 5 hónapig külön „floppyban is" gondolkodtunk, júniustól viszont már közvetlenül a CD struktúrájába épülhet be mindaz, ami eddig a floppyn volt. Szeretnénk a floppymelléklet minden jó hagyományát átörökíteni a CD-mellékletre. Arra ugyan nincs energiánk, hogy a teljes CD-t magunk állítsuk össze, de igyekszünk a OpenBlue Bt-vel szorosan együttműködve a CD-re „konvertálni" 8 évnyi floppys tapasztalatunkat. A CD-n a Lapraforgó ad helyet a cikkek elektronikus folytatásainak, illusztrációinak, de juttatunk anyagot az anyalap témáihoz kapcsolódva a CD Szerszámosláda rovatába is.

A búcsú perce

Búcsúzunk tehát a floppymelléklettől. Élt 8 évet, melynek során 4 és fél évig 5,25 collos, 360 Kbájtos ruhában lépett fel, majd 1 évre lecserélte azt 1,2 Mbájtosra, az utolsó 2 és fél évében pedig 3,5 collos, 1,44 Mbájtos köntösre. Végigkísérte az [Új] Alaplap 96 számát. Indulásakor 2 évre a Polaroidot választotta, majd kisebb-nagyobb félrelépésektől eltekintve utána végig kitartott a KAO mellett. Elmúlása szükségszerűen következett be, nem ért minket váratlanul... de azért képernyőnk sarkában látunk egy-két könnycseppet. Önök is őrizzék meg őt szeretettel emlékezetükben — és floppytartójukban.

Faklen Pál

Alapozó

Makróvírusok

Szinte hihetetlen, hogy a mindennapjainkba egyre jobban betolakodó makróvírusok alig hároméves múltra tekintenek vissza. A legelső ismert makróvírus az 1994. decemberében készült DMV volt, de ez csak jóval megírása után került ki szerzője kezeiből. A felhasználók és a nagyvilág számára az első találkozást a makróvírussal a Concept jelentette (1995 közepén), mely felbukkanása után rövid idő alatt hihetetlenül gyorsan elterjedt világszerte. Ebben elévülhetetlen érdemeket szerzett a Microsoft: egyrészt több, általa terjesztett CD-re is feltette (akaratlanul) a vírust, másrészt sokáig tagadta a létezését, majd pedig elbagatellizálta a kérdést. Amikor már nem lehetett a problémát továbbra is besöpörni a szőnyeg alá, akkor azzal nehezítette meg a vírusok ellen küzdők dolgát, hogy a Word dokumentumok szerkezetéről csak többhónapos közelharc után voltak hajlandók használható információkat kiadni.

A makróvírusok megértéséhez először a makróprogramok jellegét kell ismernünk. A makró tulajdonképpen az alkalmazásokba beépített belső parancsokból felépülő nyelv. Elsődleges funkciója, hogy a mechanikusan ismétlődő feladatokat automatizálni lehessen. Egy ilyen nyelvnek biztosítania kell a felhasználás legfontosabb parancsainak végrehajtását, azok egymás után fűzését, ha pedig ezen felül még bonyolult programozási elemeket is nyújt (elágazás, ciklus), akkor az már igazi adomány.

A Microsoft is ludas

Nos, a Microsoft olyan komplex makró-programnyelvet (WordBasic) mellékelt a Word for Windowshoz, amely lehetőségeit illetően a maga korában egyedülállónak számított. Ugyanis nemcsak a Word összes belső parancsához nyújtott hozzáférést, hanem lehetővé tette automatikusan végrehajtható programok írását, sőt a beépített Word parancsok kibővítését, testre szabását is.

Természetesen nemcsak a Word for Windows makrónyelve rendelkezik mindazokkal a lehetőségekkel, amelyek egy vírus elkészítéséhez szükségesek. Szinte minden elterjedtebb szövegszerkesztőben vagy táblázatkezelőben van ilyen makrónyelv. Ennek megfelelően készültek is AmiPro, Excel, Lotus 1-2-3 makrónyelven írt kártevők, ezek azonban annyira elenyésző számban kerültek forgalomba, hogy ismertetésünkben most nem térünk ki rájuk.

A Microsoft célja ezzel persze nem az volt, hogy a gyanútlan felhasználókat kiszolgáltassa a rosszindulatú vírusírók kénye-kedvének, hanem hogy maximális mértékben kezelhetővé tegye a Word programot.

Fontos körülmény, hogy makrókat csak sablonokhoz (template) lehet csatolni, amelyek .DOT kiterjesztésűek, megkülönböztetendő a .DOC kiterjesztésű szokványos dokumentumoktól. A Word viszont van olyan intelligens, hogy ha egy sablon esetleg .DOC kiterjesztést kap, akkor is észreveszi, hogy nem szokványos dokumentum, és minden külön tájékoztatás nélkül helyesen kezeli azt. Minden ismert MS Word makróvírus ilyen .DOC kiterjesztésű, de a sablonokra jellemző belső szerkezetű állomány. Valójában annyira minimális a különbség egy dokumentum és egy sablon között (a sablon annyival több, hogy makrókat, egyedi gombokat és menüpontokat, valamint Autotext bejegyzéseket tartalmazhat), hogy nem sok értelme van a megkülönböztetésnek, és a vírusokkal foglalkozó irodalom gyakran össze is mossa a két fogalmat.

Hatásmechanizmus

A vírusok lényegében olyan futtatható objektumok, amelyek a számítógépes környezet megváltoztatásával beilleszkednek a végrehajtási láncba, és ennek révén megsokszorozzák önmagukat. A legfontosabb eleme egy vírus létének, hogy megteremtse az aktivizálódás és a szaporodás lehetőségét. Az MS Word mindkettőt könnyedén megvalósíthatóvá teszi. A Word esetén rengeteg olyan pont van, ahol a vírus bekapcsolódhat a végrehajtási láncba. Most csak a legfontosabbakat sorolom fel.

Létezik öt előre definiált, fix nevű, ún. automatikus makró, amelyek egy-egy eseményhez tartoznak, és annak előfordulásakor kerülnek végrehajtásra. Ezeket az alábbi tábla tartalmazza:

Makrónév	Végrehajtódás feltétele
AutoExec	MS Word indítása
AutoOpen	Dokumentum nyitása
AutoClose	Dokumentum zárása
AutoExit	Kilépés az MS Word-ből
AutoNew	Új dokumentum létrehozása

Ha például egy sablon tartalmaz egy AutoClose nevű makrót, akkor ezen sablonon alapuló dokumentumok lezárásakor ez a makró automatikusan végrehajtódik. Ez a megoldás eredetileg jó célokat volt hivatott szolgálni, hiszen így például gond nélkül lehetett biztonsági másolatot készíteni minden bezárt dokumentumról. De a dolog sajnos rosszul sült el.Ha ugyanis egy mintaállományban van AutoOpen makró, akkor elegendő azt megnyitni, és huss! A Word már szabadjára is engedte a vírust, mert az a globális munkaállományba (normal.dot) másolta a vírus-makrókat és „rezidenssé" is vált, attól kezdve pedig minden további dokumentum megnyitásakor beindul az AutoOpen makró, lehetővé téve a megnyitott dokumentumok fertőzését. Az automatikus makrók végrehajtását le lehet ugyan tiltani a DisableAutoMacros paranccsal, de ez egyrészt korántsem holtbiztos megoldás, másrészt csak makróból érhető el, ami a tapasztalatlan felhasználók számára nehézséget jelent.

A vírus másik aktivizálódási lehetősége abban rejlik, hogy a Word beépített, menüből kiválasztható parancsait makrókkal át lehet definiálni. Maga az ötlet, vagyis hogy szabadon lehessen bővíteni a beépített parancsokat, mindenképpen nagyszerű, hiszen ezáltal korábban soha nem látott módon lehetett az egyedi igényeknek megfelelően alakítani, testre szabni a munkahelyeket. Sajnos azonban a vírusok is igénybe vették ezt a szolgáltatást.

Ha például létezik egy FileSaveAs nevű makró a mintaállományban (amely éppen az aktív ablakban van), akkor a Save As... menüpontot kiválasztva nem az eredeti Word parancs, hanem ez a makró hajtódik végre, és a vírus ismét vígan terjed. Ezt a felüldefiniálást viszont már semmiképpen nem lehet letiltani, és ezen a ponton teljesen védtelen marad a rendszer, ami egyértelműen a Microsoft hibája, mert valami korlátozást be kellett volna építeniük. Az még csak kisebb hiba, hogy a Word tervezésekor nem gondoltak erre, viszont az már bűn, hogy a makróvírusok megjelenése után a vírusszakértők minden kérése és figyelmeztetése ellenére sem foglalkoztak megfelelően a problémával.

A legtöbb makróvírus az automatikus makrókat használja fel a globális sablon (normal.dot) megfertőzéséhez, a menüparancsok átírását pedig a további dokumentumok fertőzéséhez. Ezenkívül van még néhány egzotikus módszer is a vírusok aktivizálódásának biztosítására.

A Gangsterz nevű vírus például azt használja ki, hogy minden makróhoz rendelhető egy billentyűkombináció, amelynek lenyomásakor a makró lefut. Normális esetben ezek a kombinációk a Ctrl és az Alt billentyűk és valamelyik betű kombinációjaként jönnek létre, a vírus ezzel szemben az „e”, illetve a szóközbillentyűk lenyomását rendeli hozzá makróinak indításához. Egy fertőzött dokumentum javításakor óhatatlanul használni kell ezek valamelyikét, így a vírus máris szabadon garázdálkodhat.

Az Underground nevű vírus azt használja ki, hogy az űrlapok szövegmezőihez makrók rendelhetők: amikor a kurzor a mezőbe lép, az odarendelt makró lefut. A vírus a dokumentum legelejére egy ilyen szövegmezőt helyez el. A dokumentum megnyitásakor a Word automatikusan a dokumentum elejére viszi a kurzort, tehát belép a szövegmezőbe, miáltal a vírusmakró lefut. Halálbiztosan, mert nincs olyan módszer, amellyel meg lehetne akadályozni a makró lefuttatását.

Árulkodó jelek

Eddig arról volt szó, hogy a Word a mit sem sejtő felhasználó háta mögött lepaktál a vírussal, esélyt sem adva felfedezésére. Nos, ez így nem teljesen igaz. Vannak árulkodó jelek. Mivel a normal.dot tartalmához hozzáadódtak a vírus makrói, kilépéskor a Word megkérdezi, hogy el akarjuk-e menteni ezt az új állapotot. Ha tudjuk, hogy magunk semmilyen vátoztatást sem hajtottunk végre, akkor erre fel kell kapnunk a fejünket. Az újabb vírusok többsége viszont már felkészült erre, és letiltja ezt a rákérdezést.

A Word önfejűsége is segítségünkre van: minden makróvírus sablonként terjed. Ez pedig a verziótól függően problémákat okoz. A Word 6.0 például csak egy meghatározott könyvtárba engedi ezeket elmenteni. Ha egy fertőzött dokumentumot megnyitunk, akkor később szintén csak ebbe, a win.ini-ben meghatározott könyvtárba menthetjük el. Ha tehát azt tapasztaljuk, hogy bárhogyan küzdünk, csak a sablonok könyvtárába tudjuk elmenteni dokumentumunkat, akkor kezdjünk határozottan gyanakodni. A Word 7.0 ezzel szemben bármilyen könyvtárba hajlandó menteni, de kifejezetten kiírja, hogy a formátum sablon. A fenti tünetek elvileg elég könnyen megkerülhetők, a ma ismert vírusoknak mégis csak elenyésző töredéke teszi ezt meg (e havi vírusunk, a CAP ebbe a kisebbségbe tartozik).

Multiplatform

Külön pikantériája a dolognak, hogy a Word makróvírusok az első, valóban platformfüggetlen vírusok, amelyek képesek különböző architektúrájú gépek között terjedni. Minden olyan operációs rendszer alatt életképesek, amelyikre elkészült az MS Word megfelelő verziója (MacOS, DOS, Windows NT, OS/2). Hogy ez nem üres fenyegetés, azt az is bizonyítja, hogy már több alkalommal észleltek Concept-fertőzést Macintosh gépeken. Gyakorlatilag mindegyik ismert Word-vírus képes más operációs rendszer alatt is terjedni (bár mindegyiket Windows alatt írták), legfeljebb néhány apróság nem működik bennük (például az ablakok színezésének megváltoztatása a Colors esetében, vagy a merevlemez formázása a trójaik esetében).

A Word különböző nyelvi verziói bizonyos mértékig gátat szabnak a szabad terjedésnek. Ezekben a változatokban ugyanis az egyes menüpontok és a hozzájuk rendelt belső parancsok neveit is lefordították. Például az angolszász változatokban meglévő FileSaveAs belső parancsnak a német verzióban a DateiSpeichernUnter felel meg, a hollandban a BestandOpslaanAls stb. Emiatt az angol verzióban megírt Concept például nem életképes német nyelvterületen. (A vírusírók persze mindent megtettek az internacionalizmus érdekében, és sorra jelentek meg a Concept változatai a német, francia, holland Word-verziókhoz).

Ne örüljünk azonban annak, hogy a kelet-albán Word védve van a vírus ellen. Sajnos az automatikus makrók minden vátozatban ugyanarra a névre hallgatnak. Mivel pedig a makróparancsok a sablonokon belül nem nevük szerint, hanem már félig lefordított 2-3 bájtos tokenekként tárolódnak (és ezek a tokenek ugyanazok minden változatban), a csupán automatikus makrókkal operáló vírusok, mint például az igen elterjedt Wazzu, gond nélkül működnek bármilyen Word-verzióban. Sőt, manapság már olyan makróvírusok is léteznek (és erre a legjobb példa a CAP), amelyek képesek alkalmazkodni az egyes nemzeti Word-változatokhoz.

A siker titka

Mi tehát a titok? Miért kerültek szinte pillanatok alatt a toplisták élére a makróvírusok?

1. Sokkal gyakrabban küldözgetünk, cserélünk dokumentumokat, mint programokat vagy floppykat, ezért a hagyományos vírusoknál jóval gyorsabban terjed szét egy fertőzés.

2. Több alkalommal még a hivatalosan terjesztett CD-kre is került makróvírus (a Microsoft legalább három CD-jén szerepelt makróvírus, kettőn a Concept és egyen a Wazzu).

3. Az e-mail levelezési szokások átalakulása miatt egyre gyakoribb a dokumentumok mellékletként való küldése, és a microsoftos levelezőprogramok (Outlook, Outlook Express, Exchange...) alapértelmezésben ezeket rögtön a Worddel nyitják meg, szélesre tárva a rendszer kapuit a Word makróvírusok behatolása előtt.

4. A WordBasic nyelvben sokkal egyszerűbb megírni egy vírust, mint Assemblyben, és gyakorlatilag a PC architektúráját sem kell hozzá alaposabban ismerni. Emiatt egy kezdő vírusíró sokkal könnyebb megír egy makróvírust, mint mondjuk egy bootvírust.

Bár a vírusszakértők a fentiekkel tisztában voltak, mégsem akadt olyan, aki két évvel ezelőtt megjósolta volna, hogy a vírusprobléma ennyire áttolódik a makróvírusokra. Most viszont már jóstehetség nélkül is megállapítható, hogy az évezred végéig bizonyára ezek a vírusok maradnak a legelterjedtebbek. Utána pedig ki tudja, mi lesz? Hátha egyszer a Microsoft (vagy valaki más!) megfogadja a szakma tanácsait, és olyan operációs rendszereket és alkalmazásokat készít, amelyek ha nem is zárják ki teljesen a vírusokat, legalább nem tárják ki előttük szélesre a nagykaput.

Szappanos Gábor

A hónap vírusa: CAP

Egy szürke kis karrierista

Egy kis bűntudattal vágok bele az e hónapra kiszemelt vírus ismertetésébe. Egyrészt a CAP nem új vírus, nem is különlegesen érdekes, nincs látványos hatása, látszatra amolyan szürke kis egér. Amikor a vírus első példánya 1997 tavaszán eljutott hozzám, teljesen érdektelennek és jelentéktelennek tűnt számomra. Az élet azonban rácáfolt elhamarkodott véleményemre, és jelen pillanatban a CAP a vírusfront főszereplője, hónapokra visszamenően vezet minden vírusgyakorisági listát. Hiába ismerjük és irtjuk, újra meg újra felbukkan, és komoly fertőzéseket okoz szerte a világon, így hazánkban is. Minek köszönhető, hogy ez az amúgy meglehetősen jellegtelen vírus ekkora karriert futott be? Erre keressük a választ.

A CAP vírus Venezuelában készülhetett, valamikor 1996 vége felé. Európába 1997 februárjában jutott el, a tavasz folyamán már komoly fertőzéseket okozott, majd nyár elején bejött Magyarországra is.

A CAP alapesetben 10 titkosított makróból áll, ezek: CAP, AutoExec, AutoOpen, AutoClose, FileOpen, FileClose, FileSave, FileSaveAs, ToolsMacro, FileTemplates. A víruskód teljes hossza 2593 bájt, és szinte a teljes egészében a CAP nevű makróban összpontosul, a többi makró gyakorlatilag csak az ebben elhelyezett eljárásokat hívja meg.

Fertőzés és rejtőzés

Amikor egy fertőzött dokumentumot nyitunk meg, vagy mentünk el, a globális sablont automatikusan megfertőzi a vírus. E folyamat során a globális sablonban lévő összes korábbi makrót (tartalmazzon az akár más vírust, akár vírusvédelmet) kitörli.

A CAP rendkívül fertőzőképes. Amint megszállta a globális sablont, a továbbiakban minden egyes megnyitott, bezárt vagy elmentett dokumentumot megfertőz. Sőt nemcsak a dokumentumokba, hanem az RTF (Rich Text Format) formátumban elmentett állományokba is befészkeli magát, azokat valójában sablonként menti el, de RTF kiterjesztéssel. Amikor egy ilyen dokumentumot megnyitunk, a Word egy szót sem szól róla, hogy a kiterjesztés és a formátum nincs összhangban, szolgálatkészen megnyitja a kérdéses fájlt. A CAP trükkje azért is hatásos, mert a vírusokkal szembeni védelem érdekében sokan dokumentumformátum helyett RTF formátumban küldözgetik egymásnak a fájlokat, és amint látható, ez a védekezési mód igen könnyen kijátszható.

A vírus sikerének egyik legfontosabb összetevője, hogy létezését és tevékenységét jól elrejti a felhasználó elől. A makróvírusok túlnyomó többsége azzal árulja el magát, hogy a megfertőzött dokumentumok más néven való elmentését (FileSaveAs) a sablon formátumának köszönhetően csak a Word által létrehozott sablonkönyvtárába engedélyezi. (A korlát csak a File|Save As parancsra él — ha a sablon már eleve máshol van elhelyezve, akkor a File|Save paranccsal mentve a helyén marad). Ezt a vírus egy ügyes trükkel megkerüli: ha ilyen mentést kezdeményezünk, akkor az új néven létrehoz egy olyan új dokumentumot, amely a régi fertőzött sablonon alapul, így annak teljes tartalma (természetesen a makrók kivételével) átkerül az új dokumentumba. Lévén ez egy közönséges dokumentum, bárhova elmenthető.

Ezután már semmi egyéb dolga nincs, mint ezt az új dokumentumot megfertőzni. A felhasználó mindebből semmit sem vesz észre. Egy kicsit lelassul ugyan a mentés folyamata, de a mai világban a Microsoft annyira beidomította a felhasználókat, hogy a sebességcsökkenést a fejlődéssel szükségszerűen együtt járó jelenségnek tekintik.

A rejtőzködés másik fontos eleme: azokat a vírusra veszélyes parancsokat, amelyekkel a makrók felfedezhetők lennének, a vírus eltávolítja a menüsorból. Ezek a parancsok a Tools|Macro (Eszközök|Makrók), a Tools|Customize (Eszközök|Testreszabás), illetve a File|Templates (Fájl|Sablonok).

A makróvírusok terjedésének egyik legkomolyabb gátja, hogy legtöbbjük csak abban a nyelvi verzióban életképes, amelyikben megírták. Ennek oka az, hogy a vírus terjedési funkcióit a legtöbb kártevő a Word egyes menüparancsainak átdefiniálásával biztosítja. Ez viszont erősen függ a nyelvi verziótól, hiszen például a dokumentum mentését az angolszász változatban a FileSave makró definiálja felül, a németben a DateiSpeichern, a franciában a FichierEnregistrer, az olaszban a FileSalva stb. Ha tehát egy vírus a FileSave nevű makróval csípte el a mentés folyamatát, akkor az a francia nyelvterületre áttérve már hatástalanná válik.

A CAP egy ötletes trükkel azt használja ki, hogy bármennyire is megváltozott a menüpontok (és így a hozzájuk rendelt belső Word parancsok) neve, a menüsorban elfoglalt pozíciójuk változatlan marad. Tehát az említett FileSave parancs minden esetben az első legördülő menüpont ötödik tagja lesz. Nem elírás! Ugyanis a menüpontokat elválasztó vízszintes vonalak is külön tagnak számítanak.

Így csak annyit kell tennie a vírusnak, hogy kiolvassa az ebben a pozícióban levő menüponthoz rendelt belső parancs nevét, és a FileSave makrót ezen a néven elmentve máris adaptálódott az idegen nyelvi környezethez. A CAP a FileOpen, FileClose, FileSave, FileSaveAs és FileTemplates nevű makróit a helyi Word változatnak megfelelő néven másolja be a globális sablonba, illetve a megfertőzött dokumentumba.

Keressük a gyenge pontjait

A vírus módszerének természetesen buktatói is vannak. Ha ugyanis a felhasználó bármilyen módon megváltoztatja a menüszerkezetet (akár egy vírusvédelmi program, akár a Word segédprogram telepítésekor), a vírus tévesen fogja a neveket kiolvasni, és ez gubancokat okozhat. Ez azonban elég ritkán fordul elő, és tudvalévő, hogy a vírusoknak, ellentétben minden tisztességes programmal, nem kell 100%-os megbízhatósággal működniük.

A CAP működtet a vírusoknál ritkán látható generációs számlálót is (erre a Concept is kísérletet tett, de programozási hiba miatt a számláló értéke, amely a hírhedtté vált bejelentkező párbeszédablakban meg is jelent, nem növekedett). Ennek értéke a normal.dot megfertőzése után növekszik meg eggyel, majd a továbbiakban minden megfertőzött dokumentumban ezt az értéket tárolja. Ha ezek a dokumentumok átkerülnek egy másik gépre, akkor ott is megfertőződik a normal.dot, és ismét növekszik eggyel a számláló. Ezzel a vírusvadászok egy kis segítséget is kapnak, mert egy intézményen belül pontosan feltérképezhető a fertőzés terjedési útvonala, sőt az is kiderül, ha egy vagy több láncszem még hiányzik, tehát tovább kell keresni.

A CAP óvatos ugyan, de néhány árulkodó jel utal a jelenlétére. A fertőzetlen dokumentumokat megnyitáskor, bezáráskor vagy elmentéskor fertőzi meg. A sűrű mentési folyamat még a tapasztalatlanabbaknak is feltűnhet. Ez ellen a vírus úgy próbál védekezni, hogy az automatikus mentési intervallumot 10 percre állítja be, ami már eleve annyira gyakori mentés, hogy abban egy-két extra nem feltétlenül szúr szemet. Továbbá bekapcsolja a gyors mentési opciót is, ezáltal kicsit megnehezíti a Word dokumentumokat nem kellő óvatossággal és szakértelemmel kezelő víruskeresők dolgát, arról nem is beszélve, hogy felgyorsítja a mentés folyamatát.

Mi a titka?

Visszatérve a cikk elején felvetett kérdésre, választ kellene adni arra, hogy miért vált ilyen sikeressé a CAP. Az okokat inkább csak találgatni lehet, de a legfontosabb tényezők talán az alábbiak:

— A vírus rendkívül szaporodóképes. Még az automatikus makrók kikapcsolása esetén is terjed, és éppen a legfontosabb tevékenységek során (mentés, nyitás, zárás).

— A változó makróállomány miatt a víruskeresők sokáig nem boldogultak vele, emiatt szabadon terjedhetett, miközben a víruskeresőkben bízó felhasználók hamis biztonságban érezték magukat.

— Alkalmazkodik a Word eltérő nyelvi verzióihoz.

— Gondosan elrejti a vírusra utaló árulkodó jeleket.

— Nemcsak a dokumentumokat, hanem a biztonságosnak hitt RTF fájlokat is megfertőzi.

Szappanos Gábor

Vírusstatisztika: súlypontáthelyeződés

E havi statisztikai morzsánk a vírusok típus szerinti eloszlását, annak időbeni alakulását szemlélteti.



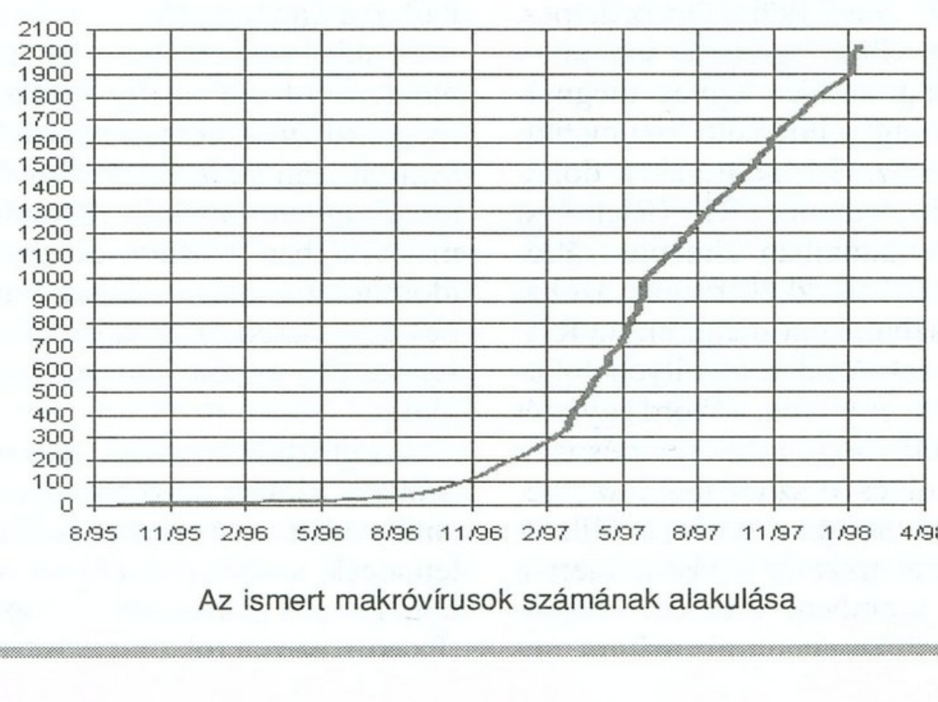
A vírusincidensek típus szerinti eloszlása

Kezdetben — a 80-as évek végen, a 90-es évek elején — körülbelül egyforma gyakorisággal bukkantak fel a program- illetve a bootvírusok. Teljesen új helyzetet teremtett azonban a Windows megjelenése és széles körű elterjedése 1992-93 tájékán.

A Windows az EXE programok szerkezetében olyan változásokat hozott, amelyekre a régebbi programvírusok nem voltak felkészítve. Ezért azután helyrehozhatatlanul megrongálták és futtathatatlanná tették a Windows programjait, így rövid úton lebukott minden régi programvírus, Windowsra megírt kártevők pedig még jó ideig nem jelentek meg, és napjainkban sem számítanak gyakori vendégnek.

A makróvírusok 1995 közepén rögtön a lista élére ugrottak. Első pillantásra meghökkentő, hogy minden átmenet nélkül jelentek meg, de világos lesz, ha belegondolunk az események menetébe. A Concept megjelenésekor a felhasználóknak fogalmuk sem volt, hogy létezhet dokumentumban fészkelő vírus, még a figyelmeztető jelek ellenére sem gyanakodtak semmi rosszra, a Microsoft viselkedése pedig a maradék éberséget is megszüntette. Mire azután pár hónap elteltével a közvélemény rádöbbent a veszély valós voltára, addigra a Concept már alaposan elterjedt világszerte.

Napjainkban, ahogy a floppyk szerepe csökken az adatmozgatásban (egyre kevésbé férnek rá az állományok egy floppyra, az új programok pedig úgyis CD-n jönnek ki), úgy zuhan a bootvírusok részaránya. A programvírusok okozta incidensek száma nagyjából változatlan, a makróvírusok pedig, „hála" a Concept és a CAP tevékenységének, egyre nagyobb szerephez jutnak.



Az ismert makróvírusok számának alakulása

A Windows jó példa...

ADO az adatbáziskezeléshez

Az objektumorientált programozástechnikának az a vonása, hogy egy újabb tulajdonság megvalósításához elegendő meg- vagy újraírni egy adott objektumnak csak az ezt megvalósító részét, jelentősen segíti a támogató környezetek terjedését. Ugyanakkor az OOP során egyszerű megoldani azt is, hogy az objektum egy adott metódusát ne csak a neve, hanem egy további azonosító alapján is elérhessük. Utóbbiakat „szabványosítva" külső eljárásként akár más objektumok is meghívhatják objektumunkat, annak eljárásait.

Ez egyrészt lehetővé teszi — az említett virtuális azonosítókat üzenetként kezelve — az üzenetvezérelt programok elkészítését. Másrészt objektumunkat bármely más program, amely alkalmas ezeknek a számára külső egységeknek a kezelésére, futásidőben is aktivizálhatja. Lehet ez olyan program is, amely az adott objektum elkészültekor még nem is létezik. De éppen készülő programunkkal, objektumunkkal mi magunk is „megszólaltathatunk" további objektumokat.

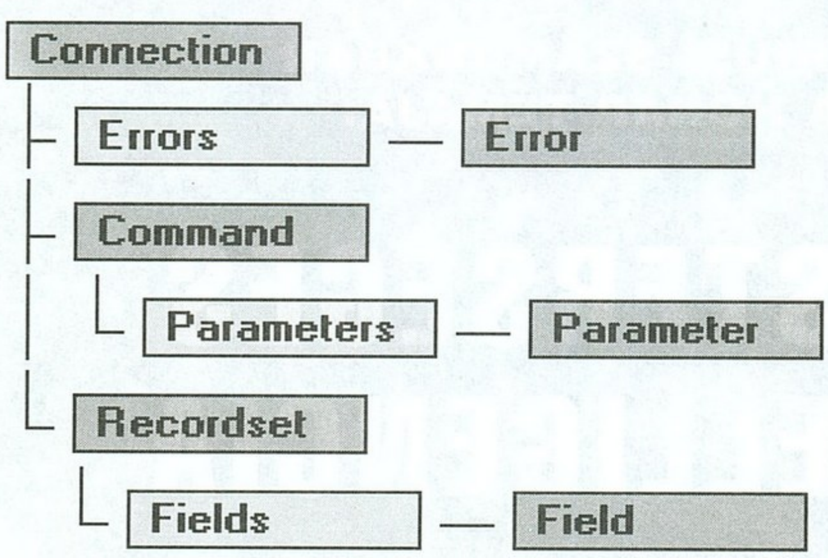
Az üzenetvezérelt és szinte teljes egészében objektumalapú programra, programrendszerre maga a Windows a legelterjedtebb példa. Az ebben felhasználható, részben szintén a Windows üzeneteit feldolgozó, leginkább azonban a Windows környezetében egyes specializált feladatokat megvalósító egységekre pedig a COM-objektumok szolgálhatnak példaként. Ezek az objektumok a Microsoft-szabványú interfészrésszel bíró, bináris állományokba fordított objektumok, amelyek kifelé megfelelnek az ezt a szabványt megvalósító modellnek (COM = Common Object Model).

Ilyen, egymással egyfajta szabványos felületen érintkező objektumokat számos programozási nyelven előállíthatunk. Készítésüket számos fejlesztőkörnyezet (C/C++, Pascal, Java, Visual Basic stb.) támogatja. Ezeknek a bináris kódba fordított objektumoknak a leggyakoribb megjelenési formái az OLE (Object Linking and Embedding) kapcsolatokat megvalósító alkalmazásobjektumok, illetve az aktív vezérlőobjektumok (ActiveX-vezérlők). Az utóbbiak az ActiveX-komponenseket kezdetben elsősorban a World Wide Web dokumentumok interaktivitásának növelésére szánták, de felhasználásuk elterjedt a 32 bites Windows-verziókban, és a különböző funkciókat megvalósító ActiveX-vezérlőknek jelentős forgalma alakult ki az Interneten, mivel fejlesztésükkel sokan sokféle programkörnyezetben foglalkoznak.

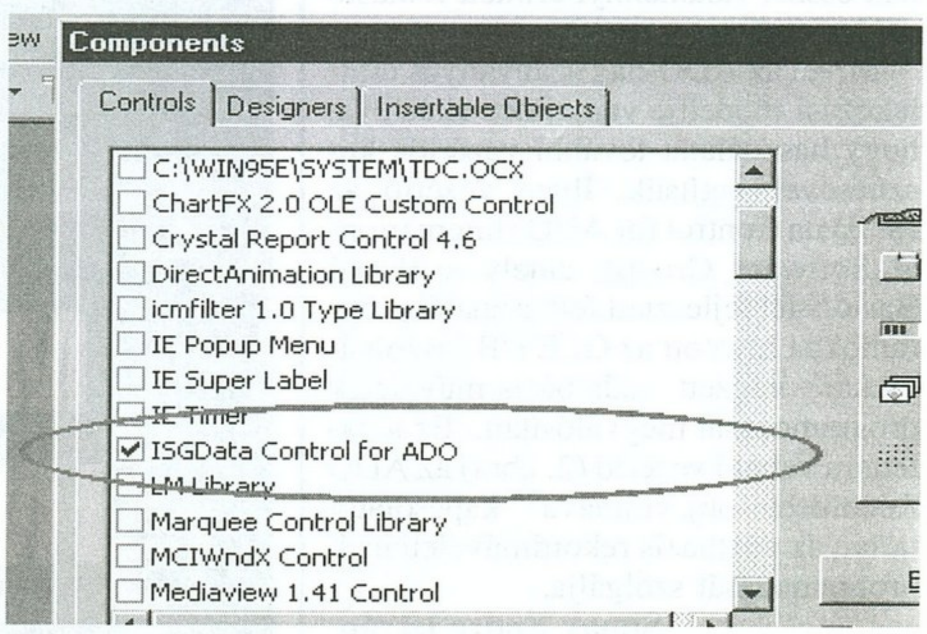
A különböző ActiveX-fejlesztésekben a Microsoft is jelentős szerepet vállal, és a fejlesztéseknek egyik szempontja az egyes programrészek, részegységek programozási felületének (API) lefedése egy-egy ilyen vezérlővel. Ezeket a későbbiekben csak egyszer kell installálni, s a későbbiekben a programfejlesztésben támaszkodhatunk a rendszer számára regisztrált objektumok felhasználására, a sokszor elég bonyodalmas API-függvények egyenként paraméterezése helyett. Az adatbáziskezelés programozásához szintén rendelkezésre állnak különböző kész objektumok. Ezek egy része az alkalmazás ablakának kialakításában részt vevő grafikus elemeket tartalmazza. A Microsoft azonban kifejlesztett egy ActiveX-alapú adatbázis-programozási objektumgyűjteményt is. Ez az ADO (ActiveX Data Objects), amely felváltani hivatott a korábban alkalmazott, szintén objektumalapú DAO és RDO rendszereket.

Az ADO elsősorban a kliensoldali programozás számára teszi lehetővé az adatbázisszerverek adatállományainak elérését (kliens/szerver és Web-programozás). Azokban az esetekben, amelyekben a korábbi ODBC adatbázisokat is lefedő OLE DB-vel elérhető adatbázisokkal dolgozunk. Ezzel elérhetjük a relációs adatbázisokat is, de kezelhetünk olyan nem relációs adatbázisokat is, mint az ADSI (Active Directory Service Interfaces) segítségével a hálózati adatokat.

Az ADO-fejlesztés célja ezeknek az adatbázisoknak a kezeléséhez szükséges programozási feladatoknak a lefe-



1. ábra



2. ábra

dése volt, miközben az ADO objektumaival biztosítani kívánták a könnyű felhasználhatóságot a készülő felhasználói programokban, illetve scriptek szintjén az ActiveX-vezérlőket fogadni képes internetes kapcsolatokban és web-szervereken. Lehetőség szerint minél kisebb merevlemez-használat és memóriaigény mellett. Az objektumgyűjteménynek jelenleg 1.5-ös verziója érhető el, de őszre várható a javított és kiegészített új, 2.0-s verzió.

Az ADO gyűjtemény az adatbázisokkal végzett munkához számos objektumot tartalmaz (1. ábra). Az ADO objektumai közül a Recordset objektum jelenti az adatbázis adataival való fő kapcsolódási felületet. Ez az objektum reprezentálja a teljes adatállományt az adatbázisban, illetve egy korábban lefuttatott parancs eredményeként kapott rekordsorozatban. Használatával megvalósítható a kurzor pozicionálása az adatbázisban, új rekord hozzáfűzése, meglevő törlése stb.

Az ADO objektumok használatakor a korábbi, szintén objektumalapú adatbázis-elérést lehetővé tevő DAO (Data Access Objects) és RDO (Remote Data Objects) objektumokkal ellentétben nem kell a teljes objektumhierarchián végigmenni, mivel ezek többsége egymástól függetlenül is létrehozható. E megoldással lehetőség nyílik rá, hogy a programozónak csak azzal az objektummal kelljen foglalkoznia, amely a tényleges feladathoz elengedhetetlen.

Így az adatbázisszerveren lefutó scriptek (ASP fájlok) segítségével és az ADO felhasználásával az adatbázissal folyamatosan kapcsolatot tartó, valóban aktív weblap is készíthető, anélkül, hogy a letöltött HTML dokumentumban az adatbáziskezelés eljárásai megjelennének. Ezek a szerveren futva a felhasználó felé egy dinamikusan előálló, az ADO révén nyert adatokkal feltöltött HTML dokumentumot állítanak össze, valamennyi érintett felhasználó számára.

Mivel az ADO-nak szabványos csatlakozási modellje van, nincs akadálya, hogy használatát további vezérlők készítésével segítsük. Ilyen vezérlő az ISGData Control for ADO (International Software Group), amely a Visual Basic 5-tel fejlesztett felhasználói programokba hivatott az OLE DB kiszolgálókkal végzett adatbázis-műveletek programozását megvalósítani. Ez a palettára vehető vezérlő (2. ábra) az ADO Recordset objektumával kapcsolatot tartva az adatbázis rekordműveleteinek programozását szolgálja.

Simay Endre István

A Mikrobazár rovatban a nem kereskedelmi célú egyéni hirdetések közlése ingyenes.

A kereskedelmi célú apróhirdetések tarifája gépelt soronként (azaz 60 karakterenként) 300 forint.

A terjedelem alapján így kiszámított összeget kérjük átutalni az Új Alaplap Kiadói Kft számlájára (OTP, 11706016–20788599), vagy feladni postai utalványon a kiadó címére (1539 Budapest, Pf. 571), és feltüntetni, hogy „Új Alaplap, apróhirdetés". A befizetést igazoló szelvény másolatát — a hirdetési szöveggel együtt — a szerkesztőséghez (a kiadóéval azonos címre) küldjék el.

Szerzői jogokat sértő szoftverhirdetéseket nem közlünk le.

Bármilyen típusú szöveg fordítását vállalom angolról magyarra, magyarról angol nyelvre, illetve vállalom kiadványok látványtervezését, szerkesztését is. Cím: Lachner Zoltán, 1195 Budapest XIX., Jáhn Ferenc u. 14/a. Telefon: 157-0308.

OBJECTS 2.0 — objektumorientált programozás CLIPPER-ben. Tájékoztató kérhető az alábbi címen: Szűcs János, 4400 Nyíregyháza, Vasvári Pál u. 37. Tel.: (42) 437-331 vagy 465-666/1382-es m.

Adatmentés CD-re, streamerre; winchesterről, floppyról. Ugyanitt beszerzési tanácsadást, hálózattervezést és programkészítést is vállalok. Cím: Kovács Lajos, 1031 Budapest III., Vízimolnár u. 10. IV/33.

Alaplapcsere, memória-, winchester- és floppybővítés a helyszínen. MegaSoft. Telefon: 295-5085.

Stúdiómban megbízhatóan, ellenőrzötten lefordítom angol, német, francia és magyar nyelvről/nyelvre műszaki és közgazdasági folyóiratok cikkeit, hardver- és szoftverleírásait. Áfás számlát állítok ki. Cím: Szász György, 1035 Budapest III., Kórház u. 25. Tel.: 168-4874.

Akarod, hogy ingyen tiéd legyen az évszázad viccgyűjteménye? Nos, ha igen, akkor írj az alábbi e-mail címek valamelyikére: qbolusyj@gold.uni-miskolc.hu vagy küldj egy üres kislemezt és egy felbélyegzett válaszborítékot a postacímemre: Kovács Gábor, 3502 Miskolc II., Pf. 83. Telefon: (20)703-438.

Keresek S3 Vision 868 vidokártyához memóriabővítést (C8256 jelű 4 db = 1 MB). Cím: Nagy Sándor, 8200 Veszprém, Lóczy u. 20/A. IV/10.

Visual Basic-ben (lehetőleg Visual Basic 5-ben) **programozót keresek** konzultáció céljából. Telefon: 329-2225 (9-17 óra között).

A Gyermekvilág '94 Alapítvány **iskolai múzeuma** adományként elfogad, vagy jelképes áron átvesz régi információs eszközöket, számítógépeket, részegységeket, tartozékokat. Az adományokról értékbecslés alapján – kérésre – adócsökkentő igazolást adunk. Cím: 8354 Karmacs, Szent Anna tér 3. Tel.: (83)372-011, este: (83)372-026.

Herdi Szabolcs nyolcadik osztályos tanuló vagyok. Nyitott gerinccel születtem, így pályaválsztási lehetőségeim nagyon lecsökkentek. Apukám 3 éve le van százalékolva, szintén gerincbetegségben szenved, anyukám munkanélküli, így anyagi körülményeik nem teszik lehetővé, hogy új számítógépet vegyenek. Akinek van egy feleslegessé vált **486-os számítógépe** és alacsony áron, részletfizetéssel el tudná adni részünkre, az ezzel sokat segítene családunk problémáinak megoldásában. Herdi Szabolcs, 6300 Kalocsa, 48-as út 27. Tel.: (78) 464-270.

„Error! Bookmark not defined."

Sajtóhibavírusok a láthatáron

Mindig szerettem a komikus sajtóhibákat, persze inkább akkor, ha azok más lapokban és kiadványokban fordultak elő. Mostani esetünkhöz legjobb analógia egy 1972-es konferencia tanulmánykötetében található. Ott a nyomdász néhány helyen szövegként írta be a szedési utasításnak szánt megjegyzést, és ennek folytán ilyetén mondatok keletkeztek: „... e gondolatához jól illeszkedett Bakonyi Sebestyén Endre nem verzál korreferátuma". A jelek szerint az elektronikus kommunikáció is kezdi kitermelni jellegzetes — és nem kevésbé groteszk — sajtóhibáit. Erről szól az alábbi nem mese.

Legutóbbi számunk Vírusőrjárat rovatában, az 53. oldalon olvasóink az alábbi mondatot olvashatták: „Az érdekes szolgáltatás az **Error! Bookmark not defined.** címen, a Dialogue Science honlapján érhető el." Hát igen, ha a szerző ezt állítja, ráadásul még érdekes is, miért ne lehetne a címzés is ilyen szokatlan. Átengedtük... bár a szóközök gyanúsak voltak.

Lapunk sajnos már épp ki volt nyomtatva, amikor megkaptuk e-mailben a Sun korrektül megszerkesztett sajtótájékozató anyagát, melynek megtekintésére kénytelenek vagyunk igénybevenni a WinWord fantasztikus szolgáltatásait, és legnagyobb meglepetésünkre abban is ilyen mondatok fordultak elő: „További információ az Interneten, a **Error! Bookmark not dedfined.** címen olvasható." Ez már több volt a gyanúsnál. Nyilvánvalóvá vált, hogy a szerzőnktől e-mailes WinWordben kapott szöveg hasonló torzuláson mehetett keresztül, és ezúttal (lapunkra egyébként nem jellemző módon) egy sajtóhiba-aknára léptünk.

Feltételezem, hogy olvasóink közül néhányan akár naivságból, akár heccből (mert ők rögtön felismerték a „leiterjakabot"), rákerestek erre a „linkre", ezért én is beírtam az URL rangra áhítozó hibaüzenetet a Netscape parancssorába. Az persze átlátott rajtam, és elegánsan átpasszolta a szöveget az Excite keresőrendszerébe, amely közölte a megnyugtató eredményt, hogy a kérdéses karaktersorozatnak 1 486 960 előfordulását találta meg. Kösz! Tehát nem vagyunk egyedül.

Miután saját 7-es WinWord-ünkben megtaláltuk a bookmark kutricáját (bár könyvjelzőt sehova nem akartunk tenni), ha őwinwördsége úgy gondolja, ám legyen kipipálva. És lássunk csodát, eltűnt a hibaüzenet. Helyén pedig ott volt a linkelésre kijelölt URL cím: aláhúzva, bekékülve. Rákattintás esetén nem ugrott ugyan sehova, nem hívott be semmilyen keresőrendszert, de a sima szövegen kívül egyebet nem is nagyon akartunk belőle kihozni.

Túl azon, hogy nem fért a fejünkbe, miért bookmark az, ami link, és hogy milyen néven kellene nevezni, ha valahová valóban bookmarkot szeretnénk tenni, érdeklődni kezdtünk a WinWord-változatok bookmarkjelöléseinek viselkedése és kompatibilitása iránt. Kutatásaink az alábbi eredményre vezettek:

A bookmarkot, pontosabban a linket a Word első értelmezésben különleges szövegrészként nyilvántartja, és a rá való hivatkozást bejegyzi a szövegen kívülre, főleg a láblécbe. Ez a speciális szöveg éppúgy lehet webcím, mint udvariassági sablonszöveg. A hivatkozás azonban csak addig működik, amíg ugyanannál a gépnél ülünk... és nem nyúlunk semmihez. Ha a névadatbázis átkerül másik könyvtárba, ha másik gépről nyomtatunk, ha ez, ha az... akkor a jelölt helyen kiírja nekünk az ominózus szöveget, hol angolul, hol magyarul... amilyen a széljárás. Megszüntetve azonban a megjelölt szöveg bookmark jellegét, ismét helyreáll a rend. Az oknyomozásban mi eddig jutottunk. Ha valaki tud pontosabb magyarázatot is, szívesen leközöljük.

Varga János

E SZÁMUNK HIRDETŐI

OOP „paszkálosoknak” és „cépluszosoknak”

Kétszer kettes ikerpár

A objektumorientált programozással kapcsolatos negyedik (a C++ nyelvről szóló) könyv megjelenésével teljessé vált a mindkét nyelvhez elvezető és az OOP-be bevezető alapvető szakirodalom és a hozzá kapcsolódó feladatgyűjtemény. A könyveket szoros rokoni szálak fűzik egybe, de ez egyiküknek sem vált a kárára.

Az objektumorientált programozás módszereivel foglalkozó hazai szakirodalommal két problémám szokott lenni. Egyrészt túlságosan elnagyolták a módszer lényegének bemutatásában (unos-untalan hajtogatják az alapelveket, de ebből még nehezen válik világossá, hogy mire jó az egész), másrészt a technikai részletek nem elég meggyőzőek, mert gyakran hiányzik mögülük a meggyőző indokolás. A nagy elvek így nemegyszer üres szavak maradnak, mert nem kristályosodik ki mögöttük a tartalom.

Magamon is tapasztalom, másokon is észrevettem, hogy procedurális gondolkodásra beállítódott agyunk más srófra jár, mint amire itt szükség lenne. Nehezen szokik hozzá, hogy most másfajta nüanszokra kellene figyelni, mint a hagyományos programozásban. Például egy hétköznapi halandónak nem könnyű átlátnia, hogy bizonyos helyzetekben mi indokolja egy-egy objektum vagy tagfüggvény ilyen vagy olyan minősítését, egy-egy kulcsszó beírását, illetve elhagyását. Vagy az istennek sem jön rá, milyen fifikával lehetne ügyesen kihasználni azokat a bizonyára határtalan lehetőségeket, amelyek benne rejlenek az új stílusú programozásban.

Talán nem én vagyok az egyedüli, aki először olvasmányain keresztül szeret eligazodni az újonnan megismert dolgokban. Persze jól tudom, hogy igazi tudást csak a gyakorlatban lehet megszerezni, de biztonságérzet is kell ahhoz, hogy az ember elhiggye: érdemes belevágni. Azért örültem meg a most ismertetendő könyveknek, mert igazolták a beléjük fektetett energiát. Úgy érzem, hogy ezzel a módszerrel végre aprópénzre lehet váltani az ígéreteket.

Elszakadó hadműveletek

Feltehetően elsősorban oktatói tapasztalataik alapján jutottak a szerzők arra a meggyőződésre, hogy az újfajta gondolkodásmód elsajátításához is a hagyományos gondolkodásmódból, a hagyományos módszerek alkalmazásából érdemes kiindulni — innen kezdve azonban nagyon következetesen, lépésről lépésre kell haladni a kitűzött cél felé. A technikai részleteket sem lehet elhagyni, mert akkor nem tudnánk mihez kötni az elveket. De az sem jó, ha a nyakába zúdítják a szegény olvasónak, mert akkor meg alaktalan, összefüggéstelen szabályhalmaz marad, amivel nem tud mit kezdeni: a kevesebb ilyenkor sokkal több lenne. Apránként vagyunk csak képesek megemészteni a dolgokat, lassan tudatosítva, hogy mire kell nagyon odafigyelni, mi az, ami itt fontos.

Az első követelmény tehát az, hogy elég apróra szét legyen trancsírozva mindaz, amit meg akarnak velünk etetni. A második meg az, hogy hagyják maguknak ízlelgetni, ne akarják erőszakkal letömni a torkunkon.

Varietas delectat

El kell ismernünk: a szerzők mindent megtettek annak érdekében, hogy se túl sokat, se túl keveset ne tálaljanak elénk. A legtiszteletreméltóbb az a szakszerűség, amellyel részeire bontották a problémákat, és az következetesség, amellyel adagolják nekünk a tudnivalókat. Módszerük egyik titka, hogy maximálisan kiaknázzák a feladatok variálásában rejlő lehetőségeket. Általában is igaz, hogy tisztábban be lehet mutatni valamilyen jelenséget, ha bonyolult helyzetben egyszerre mindig csak egyetlen paramétert változtatunk. Lényegében itt is erről van szó: a szerzők egy-egy alapfeladatból kiindulva annak apró módosítgatásaival haladnak előre. Az olvasó így könnyebben megérti az összefüggéseket, mert mindig csak egyetlen változásra kell összpontosítania a figyelmét. Gondosan összehasonlíthatja az új problémát és az új megoldást az előzővel, nem fenyegeti az a veszély, hogy valami teljesen új környezetbe, új problématérbe pottyanva esetleg nem ismeri ki magát, vagy hibás következtetéseket von le a változások szabályszerűségeiből.

Egyik ilyen lecsupaszított alapfeladat a szerzők számára egy egyszerű számológép működése. Sok programozási helyzet bemutatható ezen az alapjában egyszerű feladaton. Meg lehet mutatni a probléma megoldását C-ben, és minimális változtatással ennek OOP-

---

Benkő Tiborné — Benkő László — Poppe András:

Objektumorientált programozás C++ nyelven

ComputerBooks, 1998
378 oldal, 2464 Ft
(lemezmelléklettel)

Benkő Tiborné — Benkő László — dr. Gyenes Károly — dr. Komócsin Zoltán:

Objektumorientált programozás Turbo Pascal nyelven (7.0)

ComputerBooks, 1997
238 oldal, 1978 Ft
(lemezmelléklettel)

Benkő Tiborné — Benkő László:

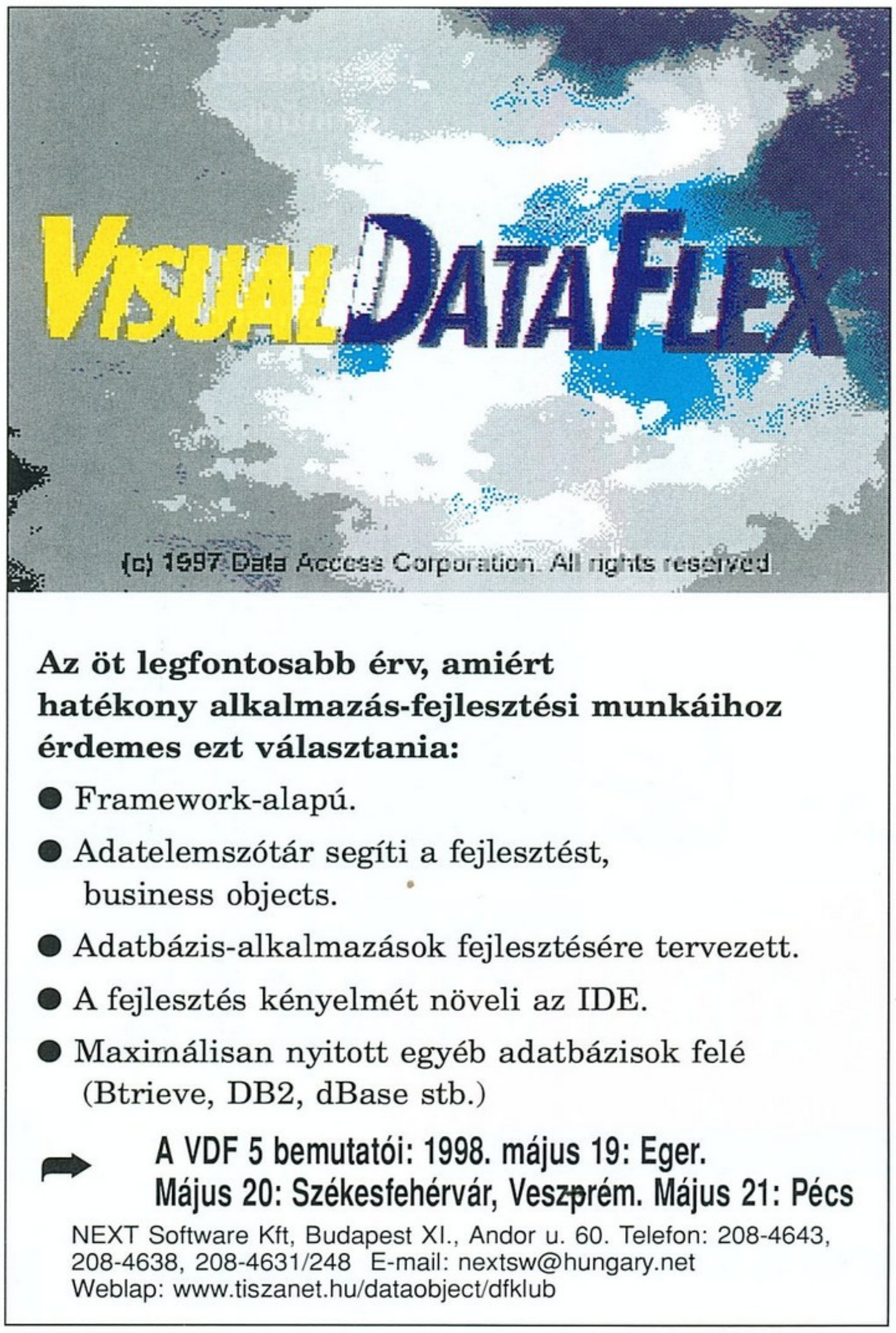
Programozási feladatok és algoritmusok C és C++ nyelven

ComputerBooks, 1997
592 oldal, 1998 Ft
(lemezmelléklettel)

Benkő Tiborné — Benkő László — dr. Meszéna Zsolt — dr. Gyenes Károly:

Programozási feladatok és algoritmusok Turbo Pascal nyelven

ComputerBooks, 1996
592 oldal, 1488 Ft
(lemezmelléklettel)



INFORMÁCIÓKÉRÉS: 26 ▶



INFORMÁCIÓKÉRÉS: 29 ▶



INFORMÁCIÓKÉRÉS: 30 ▶



INFORMÁCIÓKÉRÉS: 46 ▶



Látogasson meg bennünket az IFABO kiállítás D pavilon 402/D standján!

6400 Kiskunhalas,
Középső ipartelep 6.
Tel.: (77) 421-344
Fax: (77) 421-952
Levélcím: 6401
Kiskunhalas, Pf. 57.

HP

INFORMÁCIÓKÉRÉS: 19 ▲

WÄSCH ÖNTAPADÓ CÍMKÉK

HAGYOMÁNYOS NYOMTATÓKHOZ

30 féle méretben — 1-8 pályás elrendezéssel — 5 színben

70x36mm ➡ 0.9Ft+ÁFA 107x36mm ➡ 1.12Ft+ÁFA 120x48mm ➡ 1.28Ft+ÁFA

LÉZER- ÉS TINTASUGARAS NYOMTATÓKHOZ, FÉNYMÁSOLÓKHOZ

100 – 200 – 500 ÍVES KISZERELÉS

40 féle címke méret

LEKEREKÍTETT SARKOKKAL

Mennyiségfüggő árak: 24-29Ft+ÁFA A/4 ívenként

5 színben

Rezon TRADE

Rezon Trade Kft. 1135 Budapest, Jász u. 33-35. Tel.: 270-6026, 270-6027 Fax: 129-9038

INFORMÁCIÓKÉRÉS: 35 ▲

KeSzo Kft

1055 Budapest V., Falk Miksa u. 6.

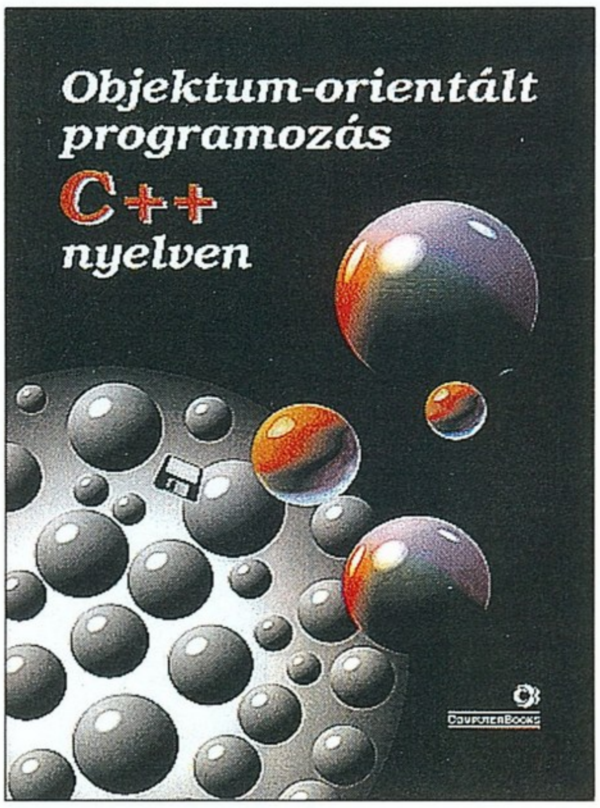
Telefon: 332-8717 Fax: 302-5136

E-mail: sales@keszo.com Web: www.keszo.com

National Geographics komplett gyűjtemény CD-n	52.000
Windows Commander 3.5 16/32bit (magyarul is)	8.800
Far 1.51 / RAR 2.02 / ARJ 2.6	7.500/7.500/14.000
Winzip 6.3 / Pkzip 2.04g	13.600/14.000
Impression (DTP sw., Quark,PM, Corel egyvelege)	98.000
F-Prot Professional	43.000
Clarion Developer 4.0/upgrade	118.000/58.000
CorelDraw 8.0 / upgrade	119.000/64.000
Hot Metal Pro 4.0	36.000
MS Frontpage 98	33.600
MS Project 98/upgrade	112.000/44.900
Norton Utilities 3.0	28.000
Norton Antivirus 4.0 angol vagy magyar	19.600
System Commander 3.x Deluxe	30.000
Norton Uninstaller	16.000
MathCAD 7.0 Professional	128.000
Procomm 4.5 Win95/NT, Internet, fax, modem, rc.	51.000
Unicode TrueType 100 betűtípus	3.900
DriveImage (FAT16/32, HPFS, NTFS)	21.000
Adobe Type Manager 4.0 deluxe for NT!	25.000
MS Office 97 magyar / upg.	112.800/47.200
MS Office 97 magyar prof. / upg.	135.800/70.000
WinFAX Pro 8.0 NT, Win95 / upg.	31.000/16.000
System Commander 3.0 Win95 (BootManager)	24.000
Partition Magic 3 (particionálás adatvesztés nélkül	23.000
Visio 5.0 Win95/NT Standard / upg.	54.000/36.000
Visio 5.0 Professional Win95/NT / upg.	110.000/52.000
Visio 5.0 Technical Win/95/NT / upg.	110.000/52.000
Photoshop 4.0 Win95/NT / upg.	179.000/64.000
Photoshop 4.0 Win95/NT magyar! / upg	179.000/69.000
NT 4.0 Server/WKS Resource Kit	34.000/16.000
Win 95 Resource Kit/ Office 97 Res. Kit	9.800/14.000
Norton Commander 1.2 Win95/NT / upg.	20.000/11.000
Clarion for Windows 2.003 / upg.	118.000/57.000
Adobe Acrobat / Corel ArtShow 7	76.000/14.400
Multikey 3.5 / upgrade	3.600/2.000
NT KEY 4.0 /upgrade előző verziókról	10.000/6.000
Adobe Illustrator 7.0 / upg.	125.000/45.000
Siearra Home +Garden	20.000
QuarkXPress 4.0 Win95/NT / PowerMAC	244.000

Áraink áfa nélkül értendők!

INFORMÁCIÓKÉRÉS: 23 ▲



beli megfelelőjét C++ nyelven. A műveletet meg lehet adni „a sorok között", inline módon. Be lehet mutatni, hogy a Művelet osztály bizonyos adattagja közvetlenül nem érhető el, ha az adattagok elérését public-ról private-ra vagy protected-re változtatjuk, bővíteni lehet viszont az osztályt egy újabb tagfüggvénnyel, hogy elérhetővé váljék az előbb „eltűnt" adattag. Lehet szemléltetni a feladaton az objektumpéldányok statikus (később dinamikus) helyfoglalását, a konstruktorok és destruktorok szerepét, a new és a delete működését, és az inicializálást különböző helyzetekben.

Meg lehet érteni ezen a példán a statikus objektumok fogalmát, sőt a több konstruktor használatának lehetőségét is, ahol az argumentumlista alapján dől el, hogy mikor melyik változatot kell aktivizálni. Jól érzékeltethető a függvények paraméterezésének megoldása a sablonok (template-ek) bevezetésével. Ez az út egyenesen vezet a típussablonnal paraméterezett osztály bevezetéséhez, ahol statikus esetben a fordítóprogram feladata eldönteni a tagfüggvények viselkedését (típusát).

Nem folytatom — a könyvben azonban nagyon szépen végig van vezetve ez a gondolatsor, szemléletesen jutva el az öröklődés és a zártság fogalmához, jelentőségéhez és felhasználási módjaikhoz, a függvények átdefiniálásához, illetve ennek speciális eseteként a C++ nyelv létező operátorai értelmezésének kiterjesztéséhez újabb jelentések hozzárendelése útján.

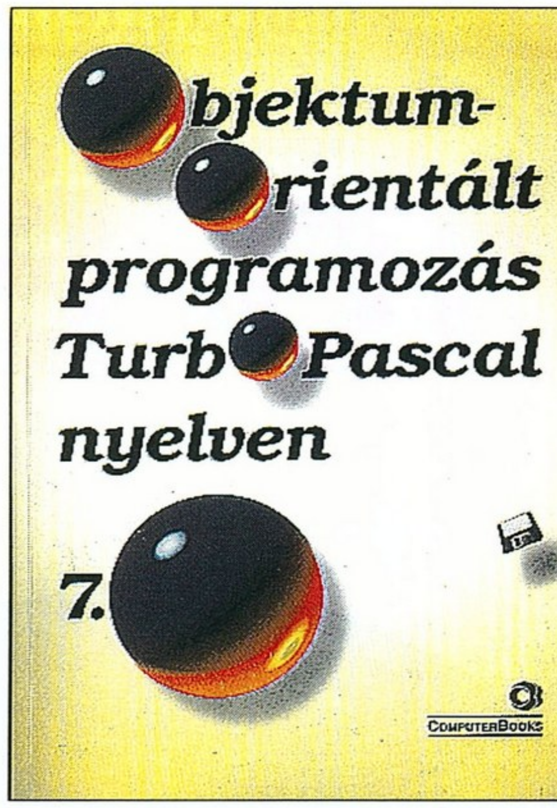
Absztrakcióóóó!

Mindez természetes módon juttatja el az olvasót a többféle irányú általánosítás lehetőségének felismeréséig. Az egyik irány a műveletek körének kibővítése újabb műveletekkel, egy másik az argumentumok körének kiegészítése más típusú (és hasonló módon, részben másként viselkedő) objektumokkal. De teljesen új tulajdonságú objektumokra is alkalmazhatóvá válik a módszer. A halmazok algebrája vagy a sztringekkel végezhető manipulációk rendszere például kétségtelenül rokonságot mutat a „szokásos" algebra és aritmetika szabályszerűségeivel, de azért egyáltalán nem triviális feladat halmazokra vagy sztringekre értelmezni a rájuk vonatkoztatható relációkat és műveleteket. Nos, a C++ nyelvű OOP-ről szóló könyv azt is bemutatja, hogy miként lehet átdefiniálni bizonyos operátorokat az OOP eszközeivel, hogy alkalmazásuk szabályai hasonlóak legyenek a Pascal nyelvben megszokott halmazelméleti konvenciókhoz.

Kettő * kettő = négy

A négy könyv koncepciója rokon (ami azért sem meglepő, mert részben szerzőik is azonosak), kidolgozásukban azonban itt-ott különbségek is vannak. A molettebb Turbo C és C++ feladatgyűjtemény teljesen alapfokon indít, és a hagyományos C-vel kezdi, de végül arra a fokra is eljut, ahová szikárabb testvére. Az „elméletibb" érdeklődésű testvér szintén nem ugrik fejest az OOP-be, hanem előbb a C++ programozásba vezet be, ugyancsak jól megtervezett lépések sorozatán keresztül. Az elméleti anyagból sem hiányoznak a példák, bár érthető, hogy több van a példatárban.

Az elméleti magyarázatok nagy része egyébként megfordítva, a szerényen csak példatárnak nevezett feladat- és algoritmusgyűjteményben is megtalálható. A C feladatok között tekintélyes helyet foglalnak el a numerikus módszerek megoldását bemutató feladatok, a lineáris egyenletektől kezdve a numerikus integrálásig, továbbá egy szép függelék egészíti ki a kötetet a Turbo Pascal és a Turbo C nyelv összehasonlításáról. Az OOP-feladatok között vannak karaktersorozatokkal foglalkozó, animációs és ütközési feladatok is. Oktatási célra olyan feladatokat is felvettek a szerzők a kötetbe, amelyek-

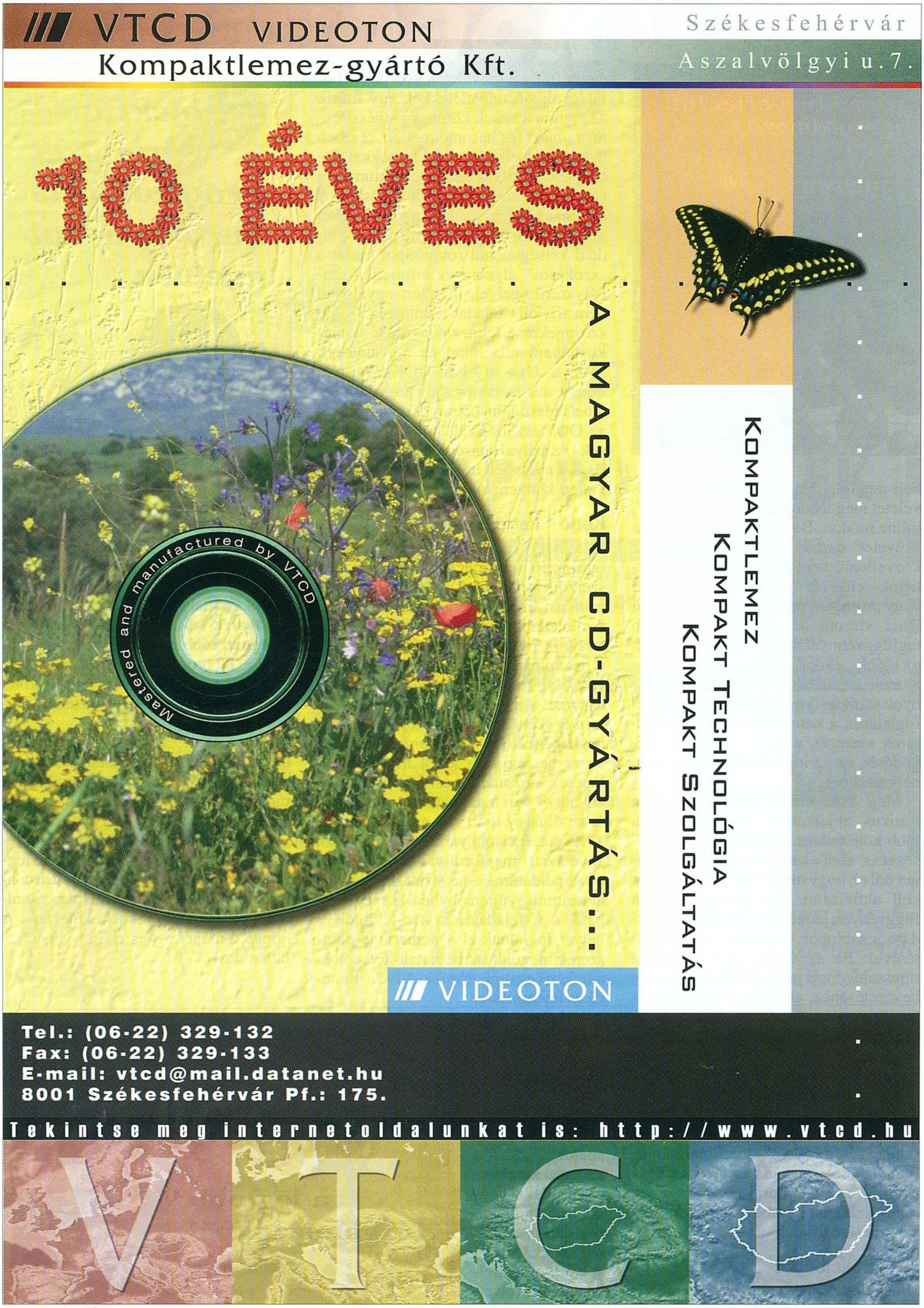


nek nincs meg a lemezmellékleten a megoldásuk.

A pascalbeli OOP-vel foglalkozó „elméleti kötetben" részben ugyanazok a feladatok szerepelnek, mint C++-beliben. A lemezen minden feladat kidolgozva megtalálható, tömörítve is meghaladja az 1 MB-ot. A példatár itt is a legegyszerűbb feladatoktól kezdve tanítja a TP programozás gyakorlatát, az OOP-vel csak ezt követően, a kötet egyik jól megtermett fejezetében foglalkozik.

Ami jelenleg magyar nyelven elérhető az OOP-ről, az indulásnak már elég, és minőségileg is színvonalasnak mondható. Az OOP szélesebb körű elterjedéshez azonban nagyon jó lenne lefordítani (vagy legalább hozzáférhetővé tenni) a külföldi szakirodalom legjavát. Főleg azokat a válogatott irodalomjegyzékeket lenne érdemes tanulmányozniuk kiadóinknak, amelyek „A hónap témája" egyes cikkei végén találhatók.

Vargha Dénes



INFORMÁCIÓKÉRÉS: 47 ▲

ELŐFIZETÉS

Az 1998/......... számtól kezdődően előfizetem

az Új Alaplap című CD-mellékletes havi számítástechnikai folyóiratot

...... példányban ❑ 1 évre, ❑ 1/2 évre.
Az éves előfizetési díj: 5880,– Ft (Ez az összeg az áfát is tartalmazza.)

❑ **Számlát** kérek (banki átutalással fizetek). ❑ **Befizetési csekket** kérek.

Név: ....

(Cég:) ....

Cím: ....

Irányítószám, helység: ....

Dátum:

/aláírás/

APRÓHIRDETÉS

Kérem, hogy az Új Alaplap következő számának Mikrobazár rovatában az alábbi szövegű apróhirdetést jelentessék meg. (A túloldalon ismertetett feltételeket tudomásul veszem.)

....

....

....

....

....

....

....

....

(Maximális terjedelem 300 betű.)

INFORMÁCIÓKÉRÉS

Az Új Alaplap mostani számában megjelent hirdetések közül az általam itt megjelölt kódszámúakhoz részletesebb információt kérek a hirdető cégektől.

01	02	03	04	05	06	07	08	09	10	11	12
13	14	15	16	17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81	82	83	84
85	86	87	88	89	90	91	92	93	94	95	96

Új Alaplap, 1998. májusi szám. Beküldési határidő: 1998. május 31.

Belföldön díjmentesen is feladható

ÚJ ALAPLAP

VI., Dózsa György út 84/b
Postafiók 571
1539 Budapest

Feladáskor kérjük bérmentesíteni!

FELADÓ:

Név: ....

Cím: ....

Helység: ....

Irányítószám: ....

Telefon: ....

❑ A hirdetés egyéni és egyedi jellegű, ezért kérem ingyenes megjelentetését. Kijelentem, hogy annak tartalma nem sérti senki szerzői jogát.

❑ A hirdetés kereskedelmi célt szolgál. Mellékelem a soronként (60 karakterenként) 300 forintnak megfelelő összeg átutalásáról az igazoló szelvény másolatát. A címzett: Új Alaplap, 1539 Budapest, Pf. 571, illetve átutalásnál az OTP 11706016-20788599 számlaszámra.

- - - - - - - - - - - - - - - - - - - -
/aláírás/

Bélyeg helye

ÚJ ALAPLAP

VI., Dózsa György út 84/b
Postafiók 571
1539 Budapest

FELADÓ:

Feladáskor kérjük bérmentesíteni!

a) EGYÉNI

Név: ....

Cím: ....

Helység, ir.sz.: ....

b) CÉGES

Név: ....

Cég: ....

Cím: ....

Helység, ir.sz.: ....

Telefon: ....

- - - - - - - - - - - - - - - - - - - -
/aláírás/

Bélyeg helye

ÚJ ALAPLAP

VI., Dózsa György út 84/b
Postafiók 571
1539 Budapest


C:\>SZEMtelenül drága
Az Ifabon is ?
B pavilon 13
ADI
www.adi.hu
Tel.: 214-8621
Fax: 214-8623
E-mail: sales@adi.hu
INFORMÁCIÓKÉRÉS: 02